Universidad Carlos III de Madrid
Escuela Politécnica Superior

INGENIERÍA TÉCNICA DE TELECOMUNICACIÓN:
SISTEMAS DE TELECOMUNICACIÓN

Proyecto Fin de Carrera

Diseño y desarrollo de un sistema de posicionamiento en interiores basado en Wi-Fi con tecnología Android

Autor: Adolfo Chico Ciprián

Tutora: Iria Manuela Estévez Ayres

Departamento de Ingeniería Telemática

Diciembre 2009

dedicado a todos los
que han estado al pie del
cañón apoyándome

Resumen

Desde hace unos años se ha producido un gran auge en las tecnologías inalámbricas y en los servicios de localización. Por ello, surgen numerosos proyectos en cada uno de estos campos, que pertenecen tanto a la rama comercial como a la de investigación. A este hecho hay que unir el nacimiento de Android, un sistema operativo que pretende ser una fuerte competencia a los sistemas existentes en el mercado.

Por ello, en este Proyecto de Final de Carrera se aborda el diseño y desarrollo de un sistema que aúne estas tres tecnologías (inalámbrica, de localización y Android), en la construcción de un sistema de localización en interiores.

El resultado de ese diseño y desarrollo es un sistema, formado por dos aplicaciones capaces de estimar la posición de un individuo en un entorno cerrado con cobertura Wi-Fi y mostrarla en un dispositivo gestionado por un sistema operativo Android. Además de la estimación de la posición este sistema puede crear un mapa de potencias, necesario para conseguir una correcta estimación de la posición. Ambas aplicaciones están preparadas para funcionar en cualquier edificio y en cualquier dispositivo que posea tecnología Android. También se ha buscado que el sistema posea una interfaz sencilla con el propósito de que pueda ser usado por cualquier persona independientemente de su nivel de conocimiento tecnológico.

Abstract

For years a great increase in the wireless technologies and the location services has taken place. For this reason numerous projects in the above mentioned fields, that belong not only to commercial branch but also to the investigation, have arisen. Apart from this fact it is necessary to complete it with the appearance of Android, an operating system that tries to be a strong competition to the existing systems in the market.

Therefore, this Project approaches the design and development of a system that combines these three technologies (wireless, location and Android) in the construction of a system of location in interiors.

The result of that design and development is a system formed by two applications. These applications are able to consider the position of a person into a building with Wi-Fi cover and show it in a device managed by Android. Besides the estimation of the position, this system creates a map in which the power levels of each signal can be seen, that are necessary to assure a correct estimation of the position. Both applications are prepared to work in any building and any device that have Android technology. It is also necessary that the system has a simple interface so that anybody can use it independently of the knowledge level.

Índice

Índice de Figuras

Índice de Tablas

Glosario

A

AES – *Advanced Encryption Standard*
ACK– *Acknowledgement*
ADT – *Android Development Tools*
A0A – *Angle of Arrival*
AP – *Access Points*
API – *Application Programming Interface*
ARM – *Advanced RISC Machines*
AVD – *Android Virtual Device*

B

BER – *Bit Error Rate*
BSS – *Base Station Subsystem*
BSSID – *Basic Service Set Identifier*
BW – *Band Width*

C

CDMA – *Code division multiple Access*
CSMA/CA – *Carrier Sense Multiple Access Collision Advoidance*
CSMA/CD – *Carrier Sense Multiple Access Collision Detection*
CTS – *Clear To Send*

D

DLC – *Dates Link Control*
DSSS – *Direct Sequence Spread Spectrum*
DTOA – *Differential Time of Arrival*

E

ESS – *Extended Service Set*
ESSID – *Extended Service Set Identifier*
ETSI – *European Telecommunications Standards Institute*

F

FCS – *Frame Check Sequence*

G

GPS – *Global Position System*
GPRS – *General Packet Radio Service*
GSM – *Group Special Mobile*
GUI – *Graphical User Interface*

H

HTML – *HyperText Mark Language*

I

IBSS – *Independent Basic Set Service*
ID – *Identifier*
IDE – *integrated Development Enviroment*
IEEE – *Institute Engenieers Electronic and Electrical*
ILS – *Indoor Location System*

J

JDK– *Java Development Kit*
JIT – *Just In Time*
JVM – *Java Virtual Machine*

L

LAN – *Local Area Networks*
LOS – *Line of Sight*
LPS – *Local Position System*
LQI – *Link Quality Indicator*
LRU – *Least Recently Used*

M

MAC – *Media Access Control*
MMU – *Memory management unit*

N

NLOS – *No Line of Sight*
NPTL – *Native POSIX Thread Library*

O

OFDM – *Othorgonal Frequency Division Multiplexing*

P

PAN – *Persinal Area Networks*
PC – *Personal Computer*
PCI – *Peripheral Compo*nen*t Interconnect*
PCMCIA – *Personal Computer Memory Card International Association*
POSIX – *Portable Operating System Interface Unix*

R

RAM – *Random Access Memory*
RF – *Radio Frequency*
RFID – *Radio Frequency Identification*
RSSI – *Received Signal Strength Indicator*
RTOF – *Round Trip Time of Flight*

RTS – *Request To Send*

S

SDK– *Software Development Kit*
SMS – *Short Message Service*
SQLITE– *Structured Query Language Lite*
SSID – *Service Set Identifier*

T

TDMA – *Time Division Multiplexing Access*
TOA – *Time of Arrival*
TOF – *Time of Flight*

U

UML – *Unified Modeling Language*
USB – *Universal Serial Bus*
UWB – *Ultra Wide Band*

W

WEP – *Wired Equivalent Privacy*
Wimax – *Worldwide Interoperability for Microwave Access*
WLAN – *Wireless Local Area Network*
WMAN – *Wireless Metropolitan Area Network*
WPAN – *Wireless Personal Area Network*

X

XML – *Extensible Markup Language*

Capítulo 1. Introducción

1.1 Contexto.

La problemática de la localización en interiores ha sido objeto de un intenso estudio e investigación durante los últimos años. Hasta ahora, ninguna de las soluciones propuestas ha conseguido el éxito alcanzado por los sistemas de localización y navegación análogos empleados en exteriores.

Las razones de que no se llegase a los objetivos propuestos eran tanto técnicas como sobre todo económicas; técnicas porque la localización en interiores plantea retos tecnológicos superiores a los de la localización en espacios abiertos, y económicas porque la mayor parte de los sistemas propuestos utilizan gran cantidad de infraestructura fija (sensores, puntos de control, estaciones base, etc.), lo que hace aumentar mucho el coste. Sin embargo, hoy en día se han reducido enormemente los costes de esta tecnología, lo cual hace que la viabilidad para instalar estos sistemas en los entornos cerrados haya aumentado.

Muchas veces en la vida diaria conocer la posición del usuario es un requisito imprescindible. Los actuales sistemas de posicionamiento global no permiten la localización con precisión de dispositivos en interiores. Debido a esto, hay una clara necesidad de desarrollar sistemas de posicionamiento o localización en interiores.

Actualmente las redes inalámbricas están orientadas a dar cobertura tanto a necesidades empresariales y particulares, como a necesidades públicas en estaciones, hospitales, universidades, en centros de ocio, hoteles, cafeterías, bares... Para muchas de estas actividades, que se realizan en estos entornos interiores, se necesita conocer la posición en la que se encuentran los sujetos.

En este proyecto se impuso como premisa utilizar la tecnología Android, ya que cada día las tecnologías *Open Source* cobran mayor fuerza y precisamente este es el caso de Android, que cada día se hace con una cuota mayor de mercado y se está convirtiendo en un fuerte competidor en el mercado tecnológico de los sistemas operativos.

1.2 Objetivos.

En este proyecto se pretende diseñar un sistema capaz de gestionar la localización de una persona en un edificio, utilizando *software Open Source* para el desarrollo de las aplicaciones y como infraestructura la red inalámbrica del propio edificio.

Para ello se desarrollará un *software* que será capaz de funcionar en un entorno real, siendo el comienzo para futuras actualizaciones y mejoras para posibles extrapolaciones a otros edificios y redes.

Este proyecto está orientado tanto para el administrador del edificio como para el usuario que desea conocer su posición, por ello el enfoque que se pretende dar es eminentemente práctico.

Además, se pretende que sea lo más universal posible y que se pueda implantar haciendo uso de los medios que ya se encuentren instalados, como sería la red Wi-Fi del edificio, evitando así realizar grandes inversiones en el montaje de una nueva infraestructura para la red inalámbrica.

1.3 Contenido.

Este proyecto empieza dando una visión del nuevo sistema operativo Android y del estado del arte de los sistemas de localización basados en radio frecuencia y en las redes Wi-Fi. Todo esto se puede ver en el segundo capítulo, el **Estado del Arte**.

Tras describir el entorno en que se desarrolla este proyecto se pasa a dar una visión de proyecto en sí. Todo esto se podrá ver en el tercer capítulo, **Desarrollo del Sistema.** Este capítulo está dividido en tres partes. La primera es la viabilidad del sistema y en ella se realiza un estudio de los requisitos que debe cumplir el sistema. En la segunda parte se ofrece el diseño de las aplicaciones de acuerdo con los requisitos expuestos en el apartado anterior. La tercera y última parte es la de la implementación, en la que se muestran los detalles de la aplicación.

El cuarto capítulo, **Pruebas y Resultados,** es el de pruebas, en el se razonan las decisiones de diseño aportando datos de que las refuerzan. Además, se presenta el resultado de la aplicación al realizar las pruebas.

El capítulo quinto es el de las **Conclusiones y Líneas Futuras,** donde se exponen las conclusiones obtenidas del diseño y la realización de las pruebas, así como una serie de pautas para posibles trabajos futuros en el sistema que se ha creado.

Para finalizar en el capítulo sexto se realiza un presupuesto de las costas del proyecto, en el que se muestran las diferentes tareas de las que se ha compuesto el desarrollo de este sistema y el tiempo invertido en cada una de ellas.

Capítulo 2. Estado del Arte.

Las aplicaciones que se desarrollan en este proyecto para el posicionamiento en interiores, se realizan sobre el sistema operativo Android, utilizando como referencia para la localización una red Wi-Fi. En este capítulo se tratará de dar una visión global de cada componente tecnológico involucrado en la realización de las aplicaciones.

En primer lugar, se verá una descripción del sistema operativo Android, que es el elemento sobre el que se desarrollarán cada una de las aplicaciones.

En siguiente lugar, se presentará un breve resumen de las distintas tecnologías que existen para estimar la localización de individuos en interiores.

Finalmente, se analizará la tecnología *Wireless* aplicada al posicionamiento de interiores y sus algoritmos de localización, que es la tecnología que se utilizará para determinar la posición del sujeto dentro del edificio. También se hará una revisión de los distintos sistemas de localización presentes en el mercado y sus algoritmos de posicionamiento.

2.1 Android.

2.1.1 Introducción.

Android es un sistema operativo además de una plataforma de *Software* basada en el núcleo de Linux. Ha sido diseñada para dispositivos móviles y actualmente esta presente en PDAs, móviles y en *Netbooks*. Es una plataforma de código abierto e inicialmente fue desarrollado por Google y luego por la *Open Handset Alliance* (liderada por la propia Google), que es un consorcio de 48 compañías de *hardware*, *software* y telecomunicaciones. Sin embargo, Google ha sido la compañía que ha

publicado la mayoría del código fuente bajo la licencia de *software* Apache, una licencia de *software* libre y código abierto a cualquier desarrollador.

Además, de incluir el sistema operativo, contiene un *middleware* y distintas aplicaciones ya implementadas. Para desarrollar nuevas aplicaciones Google proporciona el SDK de Android, que contiene todas las herramientas y APIs necesarios. El lenguaje de programación que utiliza es Java, aunque para la interfaz gráfica se puede usar XML. La razón más importante para utilizar una interfaz gráfica basada en XML es la ayuda que se obtiene gracias a las herramientas creadas para la definición de la vista de la aplicación.

Las aplicaciones de Android se ejecutan sobre la máquina *virtual* Dalvik, que es una máquina *virtual* optimizada tanto para dispositivos móviles, como para su correcto funcionamiento sobre el kernel de Linux.

En la figura 1 se puede observar una vista de alto nivel de la pila de *software* del sistema operativo de Android.

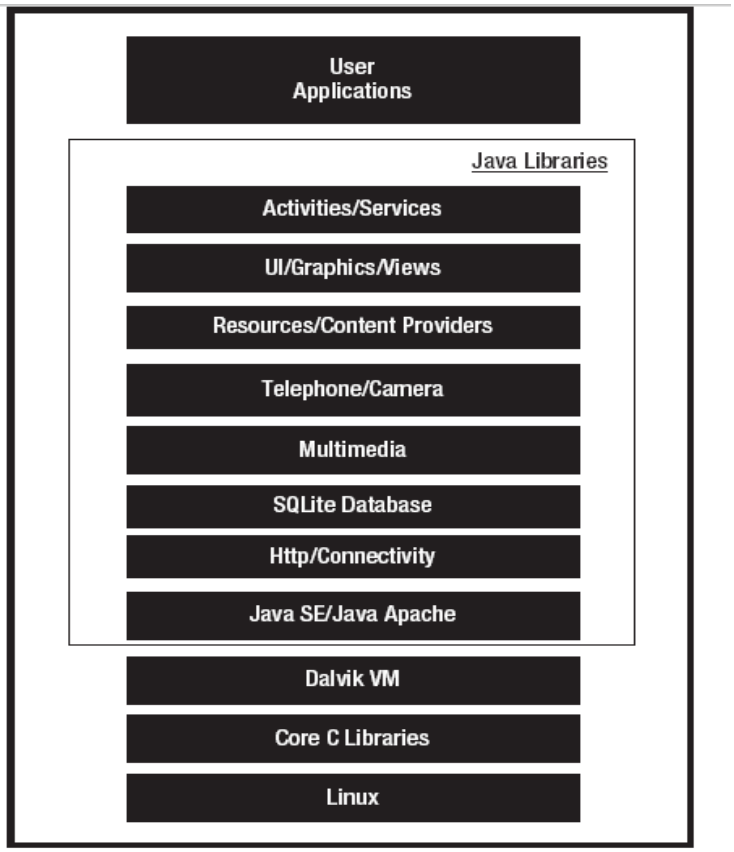


Figura 1. Vista de alto nivel de la pila de *software* de Android (obtenida de [3])

Pese a todo lo que se ha comentado anteriormente, Android no es una herramienta sólo para programadores profesionales, al ser código abierto hay una gran cantidad de

personas realizando aplicaciones, que podemos adquirir e introducir en nuestros dispositivos sin tener que realizar un gran desembolso económico, en los casos en los que haya que hacerlo.

Para acceder a muchas de las aplicaciones creadas, el grupo de Android ha creado un mercado *virtual* llamado "market", en el que se encuentra una lista con los programas que están a disposición de los usuarios. Una vez se elija uno lo descargará e instalará en el móvil automáticamente.

A continuación se va a presentar con mayor nivel de detalle las características de este sistema operativo.

2.1.2 Arquitectura.

La arquitectura de Android está formada por los siguientes bloques, que se ven en la figura 2.

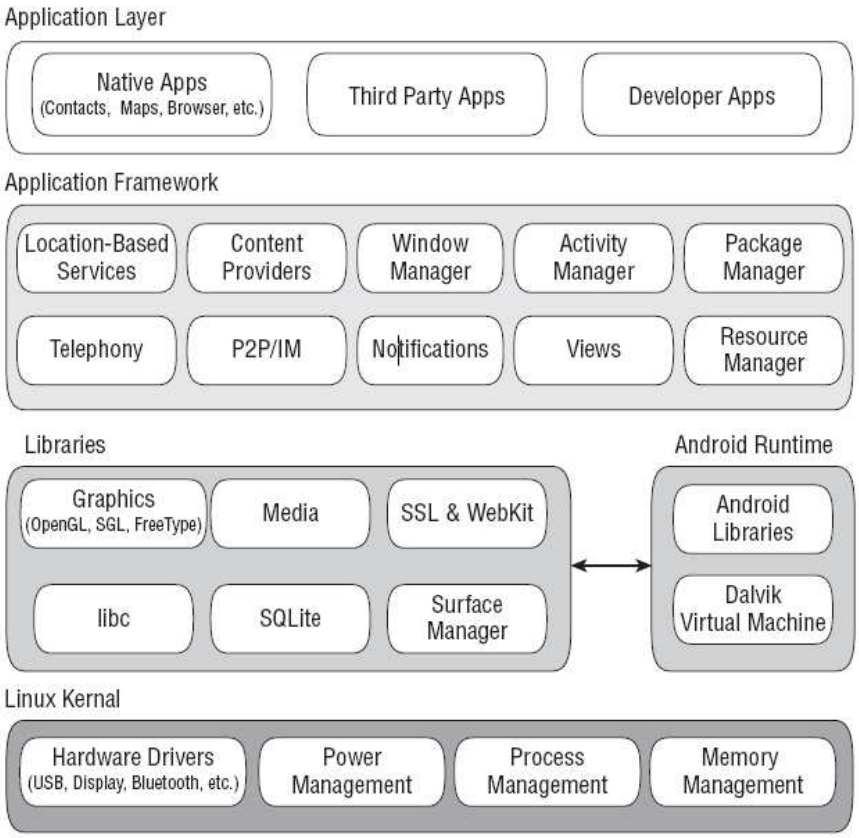


Figura 2. Arquitectura de Android (obtenida de [6])

- **Aplicaciones.** Todas las aplicaciones están escritas en lenguaje Java y pueden estar compuestas por cinco bloques: *Activity, Intent* e *Intentfilters, Broadcast*

Intents, Services y *Content Providers*. Estos bloques no tienen por qué estar todos presentes en una aplicación, se usarán los que sean necesarios para el correcto funcionamiento de la misma. Las aplicaciones base incluyen un cliente de correo electrónico, programa de SMS, calendario, mapas, navegador, contactos... entre otros.

- ***Frameworks*** (Marco de aplicaciones). Los desarrolladores tienen acceso completo a las mismas APIs del *Framework* usadas por las aplicaciones base. La arquitectura está diseñada para simplificar la reutilización de componentes; cualquier aplicación puede hacer públicas sus capacidades y cualquier otra aplicación puede luego hacer uso de ellas, siempre que no comprometa las reglas de seguridad del *Framework*. Este modo de funcionamiento permite la reutilización de código.

- **Bibliotecas**. Android incluye un conjunto de bibliotecas C/C++ usadas por varios componentes del sistema Android. Éstas se exponen a los desarrolladores a través del *Framework* de aplicaciones de Android. Las principales son: *System* C *Library* (implementación biblioteca C *standard*), bibliotecas de medios, bibliotecas de gráficos, 3D, SQLite, entre otras.

- ***Runtime* de Android**. Incluye un conjunto de bibliotecas base que proporcionan la mayor parte de las funcionalidades disponibles en las bibliotecas de Java. Cada aplicación Android crea su propio proceso, con su propia instancia de la máquina *virtual* Dalvik. Ha sido implementada de forma que un dispositivo puede ejecutar múltiples máquinas *virtual*es a la vez de forma eficiente. Esta optimización viene derivada de un uso mínimo de la memoria.

- **Núcleo – Linux.** Android depende del núcleo Linux versión 2.6 para los servicios base del sistema como seguridad, gestión de memoria, gestión de procesos, pila de red y modelo de *drivers*. El núcleo también actúa como una capa de abstracción entre el *hardware* y el resto de la pila *software*.

  El cambio fundamental de esta nueva versión de Linux es la aceptación e inclusión de gran parte del proyecto uClinux en el kernel principal. Este cambio es de gran importancia ya que la integra en el mercado *embebed*.

  En los sistemas integrados no tenemos todas las capacidades del kernel debido a limitaciones de *hardware*. Por ello la principal diferencia en estas variantes es la ausencia de MMU ("*Memory management unit*"), que hace que un sistema operativo pueda trabajar en modo protegido, embebido en el procesador.

Aunque suelen ser sistemas Linux multitarea, no tienen protección de memoria y esto hace posible que un proceso lea los datos de otros procesos. Esto hace que el sistema sea ideal para una PDA.

Otra prioridad de la nueva versión ha sido hacer que el sistema tenga una respuesta más ágil, no sólo para el usuario final sino también para aplicaciones críticas donde se requiere precisión absoluta.

Una de las mejoras definitivas en Linux 2.6 es que el kernel puede ser interrumpido a la mitad de una operación para que otras aplicaciones sigan ejecutándose, aunque habrá situaciones en que el kernel no pueda ser interrumpido.

Otro cambio importante en la versión 2.6 es la reescritura de la infraestructura del kernel para el manejo de hilos, de forma que se pueda ejecutar la Biblioteca Nativa de Hilos POSIX, NPTL (*Native POSIX Thread Library*) sobre ella. Este cambio incluye nuevos conceptos en el espacio de hilos de Linux: grupos de hilos, memoria local para hilos individuales, señales tipo POSIX...

Una de las características nuevas de Linux 2.6 más esperadas por los usuarios es la inclusión de la "Arquitectura Avanzada de Sonido Linux".

2.1.4 Aplicaciones en Android.

Todas las aplicaciones en Android están compuestas por una combinación libre de los siguientes bloques se explican a continuación:

1***. Activity o actividad*.** Son las más comunes y habitualmente es una única pantalla en la aplicación. Cada actividad es implementada como una clase que extiende a la clase base. La clase en cuestión mostrará una interfaz de usuario compuesta por *View*s que responderá a los distintos eventos que se produzcan. La mayoría de las aplicaciones están compuestas por varias pantallas, por lo que la gran parte de ellas estarán compuestas por varias actividades. El movimiento entre pantallas se logra empezando una nueva actividad. Cuando una nueva ventana se abre, la anterior queda en pausa o en una pila. El usuario puede navegar hacia atrás a través de las ventanas previamente abiertas. Es decir, básicamente es la parte visual de una aplicación.

Como ejemplo consideremos una aplicación SMS. Tiene que tener una pantalla que muestre los contactos para enviar un SMS, una segunda pantalla para escribir el mensaje y otras pantallas para leer mensajes anteriores y cambiar las

configuraciones. Cada una de esas pantallas será implementadas por una actividad. Cada vez que se inicia otra pantalla se comienza una nueva actividad.

Tal y como se puede apreciar en la figura 3 una actividad tiene un ciclo de vida muy definido, que será igual para todas las actividades. Este ciclo de vida es impuesto por el SDK de Android. Las actividades tienen cuatro posibles estados:

- Activa o en ejecución
- Pausada, cuando pierde el foco de la pantalla, pero todavía es visible y mantiene la información de estado, aunque puede ser eliminada si el sistema se encuentra en un estado crítico de memoria. Por ejemplo si el usuario está ejecutando una aplicación y recibe una llamada, la aplicación se pausará y por pantalla se mostrará la información de la llamada.
- Parada. Se da cuando ha perdido el foco de la pantalla, no es visible por el usuario pero mantiene la información de estado.
- Reiniciada. Se produce cuando una actividad estando pausada o parada se ha reiniciado y restaurado al estado anterior.

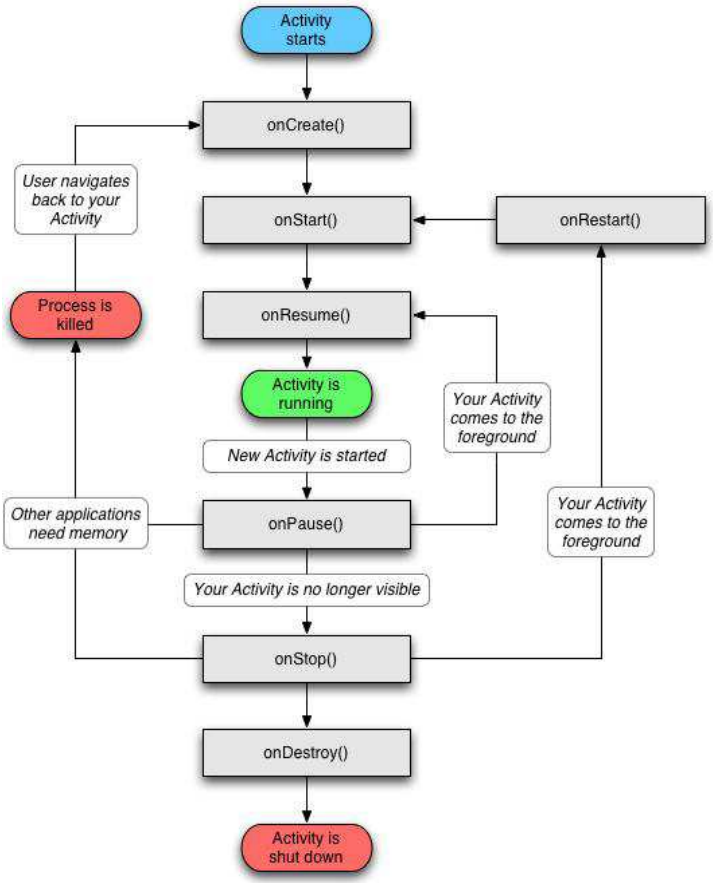


Figura 3. Ciclo de vida de una actividad (obtenida de [7])

A la hora de diseñar una aplicación en Android hay que tener en cuenta su ciclo de vida.

2. ***Intent e Intentfilters***. Son clases especiales que permiten el movimiento entre pantallas. Lo más relevante de este bloque son la acción y los datos para llevar la acción a cabo. La navegación entre pantallas se logra mediante la resolución de *Intents*. Es decir, cuando un evento de la aplicación dice "quiero hacer esto", la aplicación genera esa Intención y Android busca la más adecuada para manejarla. Muchas de las intenciones en Android están ya predefinidas, lo que facilita la labor del desarrollador, como por ejemplo controlar el click de un botón. Para afinar la búsqueda se usan los *Intentfilters*, que buscan entre todas las aplicaciones la que mejor se ajuste al *Intent*. Estos *Intentfilters* son opcionales aunque aconsejables para que la búsqueda que se realiza sea satisfactoria.

3. ***Broadcast Intent Receiver***. Son *Intents* que se utilizan cuando se quiere ejecutar algo como reacción a un evento externo.

4. ***Service.*** Se trata del código que se ejecuta sin ninguna interfaz de usuario, corriendo en segundo plano para permitir que otras actividades se ejecuten normalmente.

   Un ejemplo de este tipo de actividad es el reproductor de música, ya que se puede reproducir en segundo plano mientras se navega por las pantallas del terminal.

5. ***Content Provider.*** Todas las aplicaciones es*table*cen sus datos en distintos lugares, ya sean ficheros, bases de datos… por lo tanto tiene sentido hablar de que estas aplicaciones puedan compartir datos. Este bloque se encarga de permitir esta comunicación de datos interna.

Todas las aplicaciones contienen el *AndroidManifest*, que es el fichero encargado de controlar y decirle al sistema qué hacer con los elementos de mayor nivel creados. Las aplicaciones necesitan este fichero, ya que contiene los valores globales del paquete que se está usando, incluyendo los componentes de aplicación, la implementación de las clases para cada componente, el tipo de datos que puede manejar y cuando se pueden lanzar.

Los *Intentfilters* están incluidos en el *AndroidManifest* porque describen dónde y cuándo puede comenzar una actividad. Cuando una actividad crea un *Intent*, éste

puede tener descriptores de lo que se quiere hacer. Una vez se está ejecutando la aplicación, Android compara está información del *Intent* con los *Intentfilters* de cada aplicación, eligiendo el más adecuado para realizar la operación especificada por el llamante.

Los permisos de aplicación son una restricción que impone la máquina *virtual* Dalvik y es un mecanismo para que una aplicación que se instale no sorprenda con las cosas que hace. Cada vez que se instala una nueva aplicación, ésta nos pedirá una serie de permisos. Si no estamos de acuerdo se puede evitar la instalación y con ello evitar efectos no deseados. El *AndroidManifest* también da la posibilidad de pedir los permisos que vayamos necesitando. Además, definirá la actividad inicial que se ejecutará.

Otro elemento presente en una aplicación son las *Views.* Se trata de objetos que saben cómo dibujarse en la pantalla porque heredan las propiedades la clase *View*, que es la que gestiona la representación de elementos en la pantalla en Android. Su geometría es un rectángulo y combinando las distintas *Views* podemos crear la interfaz gráfica.

La mayor parte de las aplicaciones de Android se ejecutan en un hilo principal o *main thread* de Linux. Este hilo es creado por la aplicación en el momento que lo necesite y se mantiene hasta que no se necesite más y el sistema reclame memoria. Esto quiere decir que el tiempo de vida de una aplicación no está controlado por la propia aplicación, sino que viene determinado por el sistema. Este tiempo de vida se calcula en función de las partes de la aplicación en ejecución, así como de la importancia y el consumo de memoria que tienen.

Existe una jerarquía de importancia para los procesos en función de los componentes que se están ejecutando y el estado en que se encuentran. El sistema, en momentos de carencia de memoria, eliminará procesos guiándose por esta jerarquía, que se representa en la figura 4.

- ***Foreground/Active process***. Es el proceso que hospeda una actividad en la pantalla, con la que el usuario está interactuando. Dentro de Android hay sólo unos pocos procesos que estén dentro de esta categoría y estos procesos sólo serán eliminados cómo último recurso. Si esta fuera la situación, el dispositivo probablemente alcanzó un "*memory paging state*" y esta acción es necesaria para mantener la interfaz del usuario con capacidad para responder al usuario.

- ***Visible process***. Se trata del proceso que contiene un *Activity* que está visible en la pantalla, pero no en primer plano. Ocurre cuando se muestra un *Foreground Process* por pantalla permitiendo al visible *process* estar en un

segundo plano. Es decir, se están ejecutando dos procesos simultáneamente, pero por pantalla solo se muestra el *Active process*.

- ***Service Process***. Son procesos que no se ven directamente por el usuario, pero sí que se están ejecutando.

- ***Background process***. Son procesos que mantiene una actividad pero no es visible por el usuario. La diferencia que existe con el *Service process* es que no inciden de manera directa en la actividad que se está realizando por parte del usuario.

- **Empty *process***. Se trata de un proceso que no tienen ningún componente activo y la única razón que los mantiene funcionando es que mejoran el tiempo de arranque la siguiente vez que se use. Es decir, el proceso está abierto a la espera de ser reiniciado. De esta manera, se evita abrir e iniciar el proceso de nuevo.

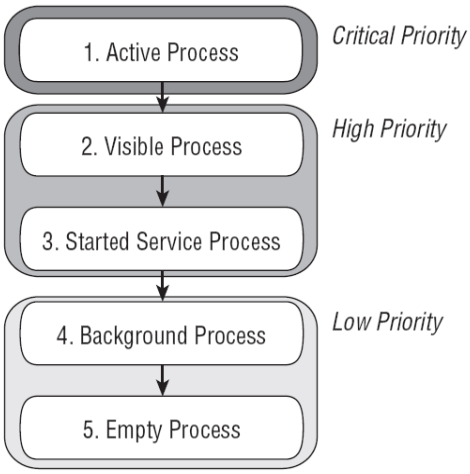


Figura 4. Nivel de prioridad (obtenida de [6])

Para los *Background process* existe una lista llamada LRU (*Least Recently Used*). En función de esta lista se van eliminando los procesos; los primeros que se eliminan son aquellos que llevan más tiempo sin usarse. Así el sistema se asegura de mantener vivos los procesos que se han usado recientemente.

El punto de entrada de una aplicación *normalmente* será una actividad (es decir, la pantalla que vemos cuando ejecutamos la aplicación). Esta actividad se compone de dos elementos: la interfaz (que estará en res/*layout*) y el código fuente (que estará en src).

2.1.5 Dalvik (*Virtual Machine*).

Google ha pasado mucho tiempo pensando en la optimización de diseños de baja potencia para pequeños dispositivos. Las previsiones dicen que estos dispositivos estarán a la par de sus homólogos de escritorio en cuanto a memoria y a velocidad dentro de diez años. También tienen un poder limitado para el cálculo, la memoria RAM total de un dispositivo móvil está en torno a los 192 MB y su espacio disponible para las aplicaciones puede ser tan sólo 20 MB (sin incluir memorias externas)

Por ello, se requieren teléfonos que sean diseñados para lograr la optimización de todos sus recursos. Si nos fijamos en la lista de paquetes en Android, podemos ver que son completos y extensos en número. Según Google, las bibliotecas del sistema pueden utilizar como mucho 10 MB, incluso con su JVM optimizado. Estos problemas llevaron a Google a reconsiderar la aplicación JVM estándar en muchos aspectos.

La figura clave para la implementación de esta JVM es Dan Bornstein, quien escribió el DalvikVM y la llamó así basándose en el nombre de una ciudad islandesa.

Primero, la máquina *virtual* de Dalvik toma los archivos generados por las clases Java y los combina en uno o más archivos ejecu*table*s Dalvik (. dex). Reutiliza la información duplicada por múltiples archivos .class reduciendo así la necesidad de espacio (sin comprimir) a la mitad de lo que ocuparía un archivo .jar.

En segundo lugar, Google ha mejorado la recolección de basura en la máquina *virtual* de Dalvik, pero ha preferido omitir un *just-in-time* (JIT), en esta versión por lo menos. La empresa justifica esta elección diciendo que muchas de las bibliotecas centrales Android, incluyendo las bibliotecas de gráficos, están implementadas en C y C++. Del mismo modo, Android proporciona una biblioteca de C optimizada para acceder a la base de datos SQLite, pero esta biblioteca está encapsulada en un nivel superior del API de Java. Dado que la mayoría del código del núcleo se encuentra en C y C++, Google argumentó que el impacto de la compilación JIT no sería significativo.

Por último, la máquina *virtual* de Dalvik utiliza un tipo diferente de montaje para la generación del código, en el que se utiliza los registros como las unidades primarias de almacenamiento de datos en lugar de la pila.

Hay que señalar que el código ejecu*table* final en Android, como resultado de la máquina *virtual* de Dalvik, no se basa en el *bytecode* de Java, sino que se basa en los archivos .dex. Esto significa que no se puede ejecutar directamente el *Bytecode* de Java, hay que comenzar con los archivos .class de Java y luego convertirlos en archivos .dex.

2.1.6 Interfaz de usuario.

La interfaz de usuario se define en los archivos XML del directorio res/*layout*. Cada pantalla tendrá un XML diferente.

Diseñar una pantalla mediante código Java puede ser tedioso y poco eficiente. Sin embargo, Android soporta XML para diseñar pantallas. Android define un montón de elementos personalizados, cada uno representando a una "subclase" específica de *View*. Se pueden crear pantallas de la misma manera que se diseñan ficheros HTML. Cada fichero describe una única pantalla, un *layout*. Éste puede contener a otros elementos y así conformar la vista de la pantalla.

Para parte de la interfaz de usuario Android introduce una nueva terminología, la cual analizaremos a continuación.

- ***Views*:** se trata de una estructura de datos que permite es*table*cer el *layout* y el contenido de un área rectangular (la pantalla). Además, puede manejar medidas, focalizar elementos, poner una barra de *scroll*, etc. Funciona como base para *widgets* (conjunto de "subclases" ya implementadas en Java que permiten mostrar elementos de pantalla interactivos).

- ***Viewgroups*:** son objetos cuya función es almacenar y manejar conjuntos de objetos *View* y *Viewgroup* hijos.

Se puede ver la jerarquía de estos elementos en el árbol que se muestra en la figura 5:

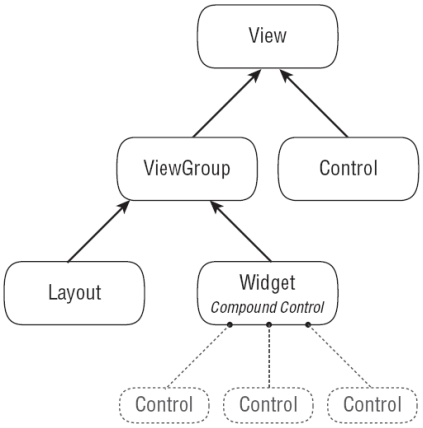


Figura 5. Jerarquía de la clase *View* (obtenida de [6])

Los *View*s y *Viewgroups* han de estar contenidos en algún elemento. Estos elementos son los *layout*s, que contienen otros elementos presentes en una vista. Dentro de cada *layout* podemos poner todos los elementos necesarios, incluidos otros *layout*s. Así

conseguiremos estructurar la pantalla de la manera deseada. Existen una gran variedad de *layout*s, en función de su posicionamiento en la pantalla:

- ***Linear Layout***: apila a sus hijos verticalmente. Sus hijos son los *view*s, *viewgroups* u otros *layout*s. En la figura 6 se observa la composición de la pantalla y el código que la genera a continuación.

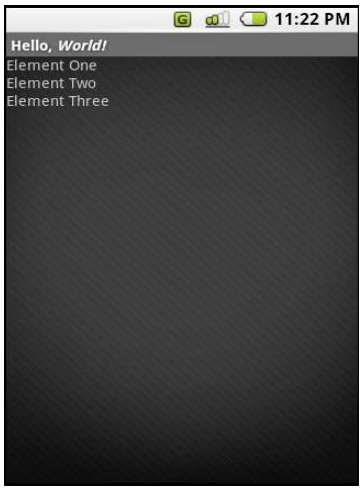


Figura 6. Vista de un *Linear Layout*.

```
<LinearLayout xmlns:android=
"http://schemas.android.com/apk/res/android"
android:orientation="vertical"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>
<TextView
android:layout_width="wrap_Content"
android:layout_height="wrap_Content"
android:text="Element One"
/>
<TextView
android:layout_width="wrap_Content"
android:layout_height="wrap_Content"
android:text="Element Two"
/>
<TextView
android:layout_width="wrap_Content"
android:layout_height="wrap_Content"
android:text="Element Three"
/>
</LinearLayout>
```

- ***Relative Layout***: en este caso todos los elementos van colocados en una posición relativa a otro. Aquí podemos jugar con las distancias entre elementos en la pantalla. La distancia se mide en píxeles. En la figura 7 se observa la composición de la pantalla y el código que la genera a continuación.

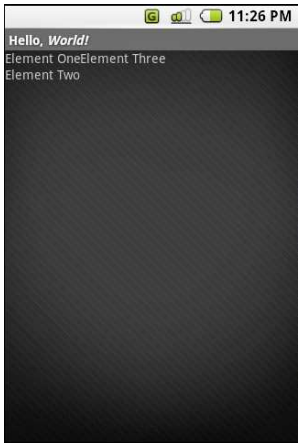


Figura 7. Vista de un *Relative Layout.*

```
<RelativeLayout xmlns:android=
"http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>
<TextView
android:id="@+id/EL01"
android:layout_width="wrap_Content"
android:layout_height="wrap_Content"
android:text="Element One"
/>
<TextView
android:id="@+id/EL02"
android:layout_width="wrap_Content"
android:layout_height="wrap_Content"
android:text="Element Two"
android:layout_below="@id/EL01"
/>
<TextView
android:layout_width="wrap_Content"
android:layout_height="wrap_Content"
android:text="Element Three"
android:layout_toRight="@id/EL02"
/>
</RelativeLayout>
```

- ***Absolute Layout:*** coloca los elementos en posiciones absolutas de la pantalla, teniendo en cuenta que la posición (0,0) es el extremo superior izquierdo de la pantalla. En la figura 8 se observa la composición de la pantalla y el código que la genera a continuación.

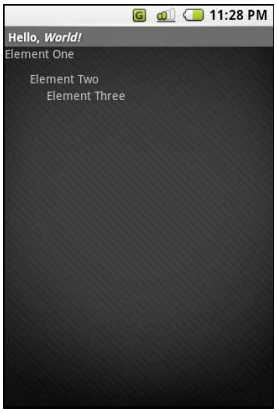


Figura 8. Vista de un Absolute *Layout.*

```
<AbsoluteLayout xmlns:android=
"http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
>
<TextView
android:layout_width="wrap_Content"
android:layout_height="wrap_Content"
android:text="Element One"
/>
<TextView
android:layout_width="wrap_Content"
android:layout_height="wrap_Content"
android:text="Element Two"
android:layout_x="30px"
android:layout_y="30px"
/>
<TextView
android:layout_width="wrap_Content"
android:layout_height="wrap_Content"
android:text="Element Three"
android:layout_x="50px"
android:layout_y="50px"
/>
</AbsoluteLayout>
```

- ***Table Layout***: permite colocar los elementos como si se tratase de una tabla. En la figura 9 se observa la composición de la pantalla y el código que la genera.

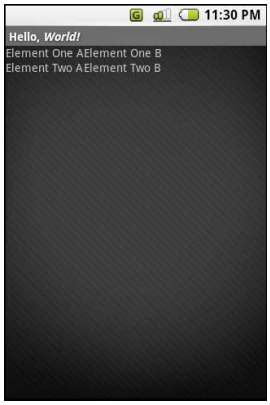


Figura 9. Vista de un *Table Layout.*

```
<TableLayout xmlns:android=
"http://schemas.android.com/apk/res/android"
android:layout_width="fill_parent"
android:layout_height="fill_parent">
<TableRow>
<TextView
android:layout_width="wrap_Content"
android:layout_height="wrap_Content"
android:text="Element One A"
/>
<TextView
android:layout_width="wrap_Content"
android:layout_height="wrap_Content"
android:text="Element One B"
/>
</TableRow>
<TableRow>
<TextView
android:layout_width="wrap_Content"
android:layout_height="wrap_Content"
android:text="Element Two A"
/>
<TextView
android:layout_width="wrap_Content"
android:layout_height="wrap_Content"
android:text="Element Two B"
/>
</TableRow>
</TableLayout>
```

En los siguientes árboles representados por las figuras 10 y 11 se pueden ver las jerarquías de los *Layout* y de la clase *View*, lo que ayudará a entender cómo organizar los elementos de la interfaz gráfica.

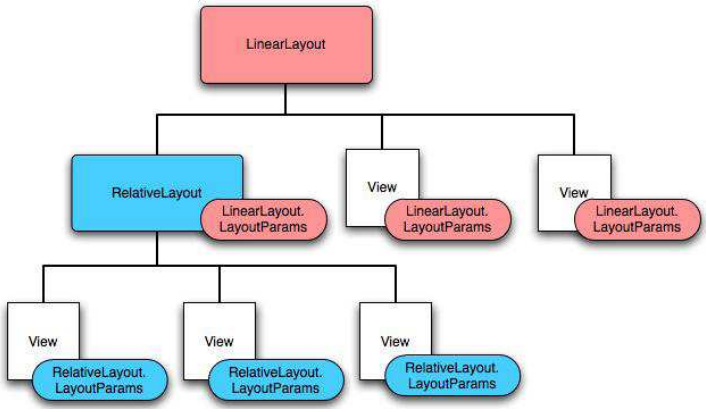


Figura 10. Posiciones relativas de los nodos hijo (obtenida de [7])

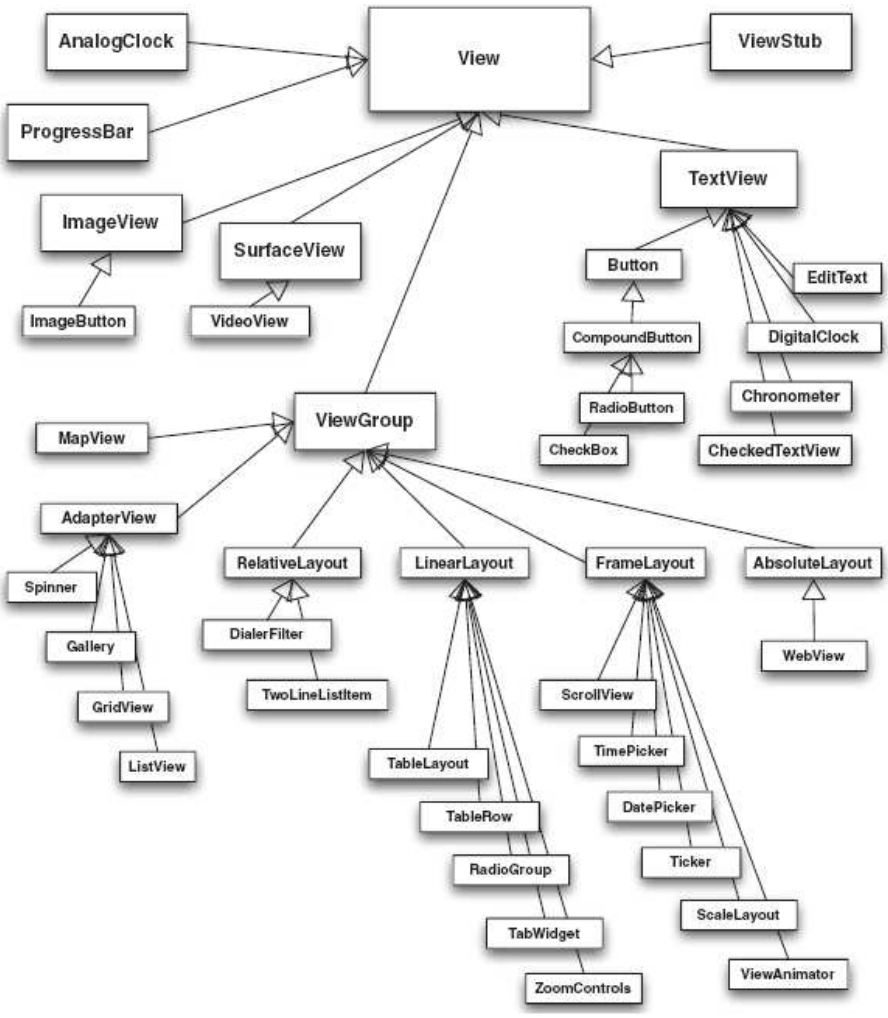


Figura 11. Árbol de jerarquía de la clase *View* (obtenido de [4])

2.1.7 Modelo de seguridad

La seguridad en Android abarca desde el despliegue hasta la ejecución de la aplicación. Con respecto a la implementación, las aplicaciones de Android tienen que estar firmadas con una firma digital para poder instalarlas en el dispositivo. Con respecto a la ejecución, cada aplicación Android se ejecuta dentro de un proceso separado. Cada uno de estos procesos tiene un ID de usuario único y permanente (asignado en el momento de la instalación). Esto impone un límite al proceso porque evita que una aplicación tenga acceso directo a los datos críticos para el correcto funcionamiento del terminal. Por otro lado, Android define un modelo de declaración de permisos protegiendo así los datos críticos o sensibles tales como la lista de contactos.

Como se ha apuntado anteriormente, Android requiere que las aplicaciones sean firmadas con un certificado digital. Uno de los beneficios de este requisito es que una solicitud no puede ser actualizada con una versión que no fue publicada por el autor original.

Las aplicaciones se firman con un certificado digital. Un certificado digital contiene toda la información acerca del creador: nombre, dirección, etc. Los atributos más importantes de un certificado digital son la firma y la clave pública o privada. Hay que tener en cuenta que el uso de certificados digitales además de ser para archivos .APK, también se puede utilizar con otros fines como la comunicación cifrada.

La firma de aplicaciones Android requiere tres cosas: un certificado digital, un archivo. APK y una herramienta o programa que sepa cómo aplicar la firma del certificado digital para el archivo.apk.

Realización de comprobaciones de seguridad en tiempo de ejecución.

La seguridad en Android durante el tiempo de ejecución sucede a nivel de proceso y a nivel de operación. En el nivel de proceso Android evita que una aplicación tenga acceso directo a los datos de otra aplicación. Esto se consigue mediante la ejecución de cada aplicación dentro de un proceso diferente y bajo un único y permanente usuario. A nivel operativo Android define una lista de las características y los recursos protegidos. Para que un programa pueda acceder a esta información hay que agregar una o más solicitudes de permiso en el archivo AndroidManifest.xml.

Seguridad de los procesos.

A diferencia de a lo que se está acostumbrado, durante nuestro uso cotidiano del PC, donde la mayoría de las aplicaciones se ejecutan bajo el mismo identificador, en

Android cada aplicación se ejecuta en general con un identificador propio y único. Al realizar la ejecución de cada aplicación bajo un ID diferente, Android crea una capa de aislamiento alrededor de cada proceso. Esto evita lo que ya se ha comentado anteriormente, el acceso a datos de otras aplicaciones. Aunque cada proceso tiene un límite alrededor de él, el intercambio de datos entre aplicaciones evidentemente es posible, pero tiene que ser explícito. En otras palabras, para obtener datos desde otra aplicación hay que pedirlos usando los recursos adecuados (petición de permisos en el *AndroidManifest*). Todas estas facilidades proporcionan métodos para compartir información entre aplicaciones, pero haciendo peticiones explícitas, ya que no se tiene acceso a la base de datos subyacente, a los archivos, etcétera.

Android define un régimen de autorizaciones destinadas a proteger los recursos y funciones en el dispositivo. Por ejemplo, las aplicaciones, por defecto, no pueden acceder a la lista de contactos, hacer llamadas de teléfono, etc. Para proteger al usuario de códigos maliciosos, Android pide las autorizaciones pertinentes para poder utilizar recursos protegidos. En el momento de la instalación, el instalador APK concede o niega los permisos solicitados sobre la base de la firma del archivo. APK del usuario. Si el permiso no se concede, cualquier *Intent*o de ejecutar o acceder a la función asociada va a provocar un error de permisos.

2.2 Sistemas de localización de interiores basados en red RF.

2.2.1 Introducción.

La Real Academia de la Lengua Española define localización como la determinación del lugar donde se encuentra alguien o algo. Esta determinación conlleva unas relaciones espaciales y abstractas entre los objetos.

El GPS es la solución casi universal a todos los problemas de localización precisa y rápida en cualquier punto del planeta. Sin embargo, este sistema no funciona en interiores, con lo que aparece un nuevo concepto, LPS (*Local Position System*). De las varias posibilidades tecnológicas para el diseño de LPS, las basadas en señales de RF experimentan un gran auge en la actualidad.

Este auge se debe a que las tecnologías inalámbricas han sufrido una fuerte eclosión en el mercado tecnológico desde hace unos años. Un claro ejemplo de este aumento en su uso, es que estas tecnologías son de uso cotidiano por la sociedad:

- Redes personales (PAN: *Personal Area Networks*). Bluetooth, ratones y teclados inalámbricos.

- Redes locales (LAN: *Local Area Networks*). Wi-Fi.

- Integración y convergencia con los sistemas celulares ("globales"), como GPRS y móviles de 4ª generación (*All IP*).

Gracias a la masificación del uso de estas tecnologías, están desarrollándose estándares, haciendo que surjan nuevas aplicaciones tecnológicas como la conectividad continua (voz, audio, vídeos digitales), las redes de sensores inalámbricos, los sistemas de rescate y emergencias, los servicios basados en la posición…

Todo este aumento del interés por los sistemas de posicionamiento en interiores mediante el uso de Wi-Fi se puede ver reflejado en la siguiente gráfica, que muestra el aumento de las publicaciones en esta materia durante los últimos años. En la figura 12 se muestran todas las publicaciones existentes en Google y la información ha sido obtenida del **Máster Oficial en Sistemas Electrónicos Avanzados. Sistemas Inteligentes.**

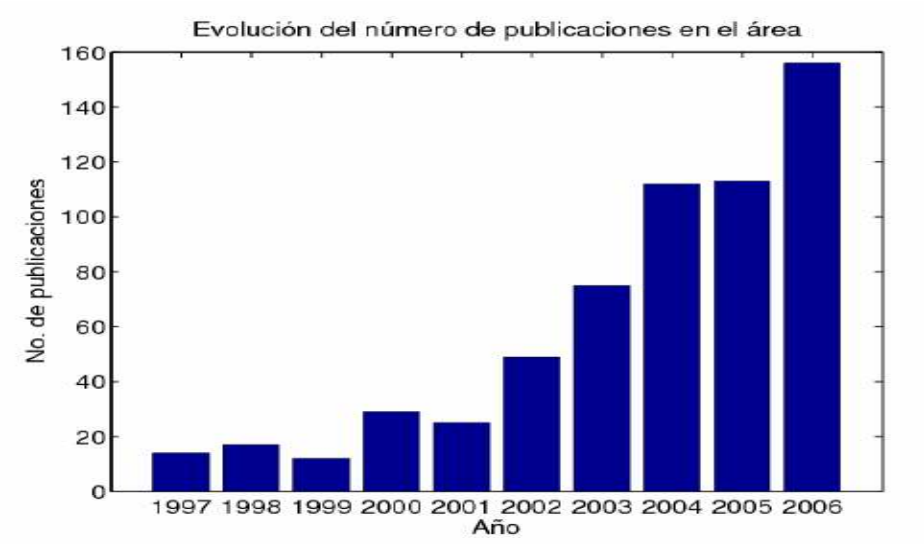


Figura 12. Gráfico de aumento de las referencias al posicionamiento en interiores en Google (obtenida de [11])

2.2.2 Características de diseño de los sistemas de posicionamiento basados en RF.

Existen muchas posibilidades en el diseño de los sistemas:

a) *Según la tecnología física que los sustenta*. Es decir, se puede obtener mayor o menor precisión en función del sistema que se use. En el gráfico de la figura 13 se pueden ver las distintas tecnologías, su ámbito de trabajo y la precisión que se obtiene.

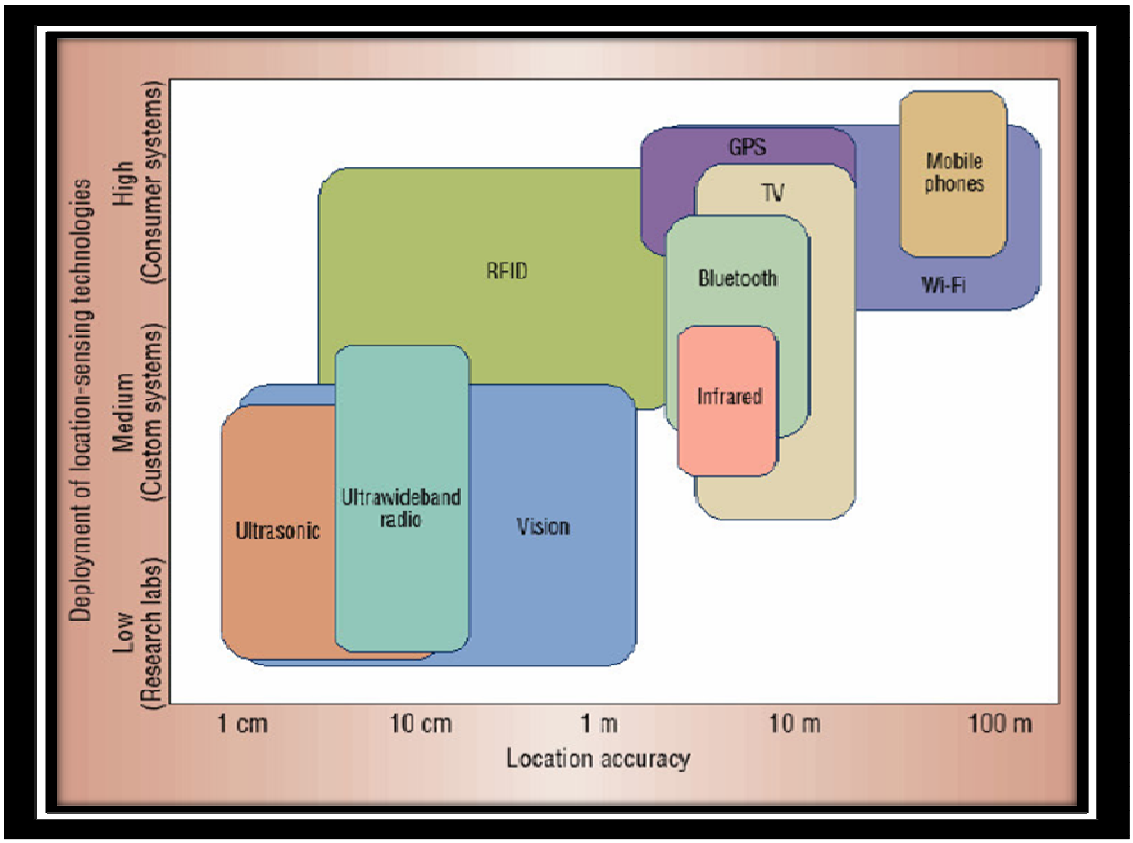


Figura 13. Clasificación de las tecnologías RF (obtenida de [11])

b) *Según la magnitud física observable.* En este caso podemos medir distintos aspectos de la señal, que nos ayudaran a conocer nuestra posición.

- **Basados en medida de rangos.** Para efectuar la medida de rangos nos basamos en la medida matemática de trilateración, que consiste en usar la geometría de triángulos de forma análoga a la triangulación. La diferencia en este caso es que se usan medidas de ángulo junto con al menos una distancia conocida para calcular la localización del sujeto. La

trilateración usa las localizaciones conocidas de dos o más puntos como referencia y la distancia medida entre el sujeto y cada punto de referencia.

En la figura 14 se observa cómo se realiza el cálculo de la posición para este tipo de tecnología.

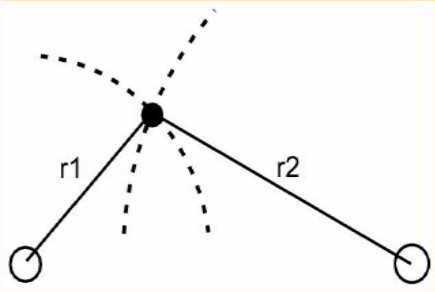


Figura 14. Intersección de señales basadas en medidas de rangos (obtenida de [11])

En función del tiempo que se mida, se puede usar una figura geométrica distinta para realizar la intersección. En la tabla 1 se observan que tipo de intersección hay que resolver en función de la situación y el observable.

Observable	Situación	Intersección de
TOA, TOF *(Time of Arrival/Flight)*	Sincronía entre móvil y balizas	Esferas
DTOA *(Differential Time of Arrival)*	Sincronía entre balizas, pero no con el móvil	Hiperboloides
RTOF *(Round-trip Time of Flight)*	Sin sincronía	Esferas

Tabla 1. Representación de los observables, la situación y la intersección de señales basadas en medidas de rangos (obtenida de [11])

- **Basados en medidas de ángulos**. El método matemático para realizar las medidas correspondientes es la triangulación, que se basa en el ángulo de llegada de la señal. En la figura 15 se observa una representación de la intersección de las señales utilizando esta tecnología.

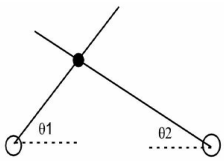


Figura 15. Intersección de señales basadas en medidas de ángulos (obtenida de [11]).

En este caso tan solo observamos el ángulo de llegada de la señal. La tabla 2 muestra en resumen el observable, la situación y el tipo de intersección.

Observable	Situación	Intersección de
AOA *(Angle of Arrival)*	No es necesaria sincronía entre móvil y balizas Antenas direccionales (GSM) Arrays con medidas de diferencias de fase	Rectas / planos

Tabla 2. Observable, situación y tipo de intersección para señales basadas en medidas de ángulos (obtenida de [11])

- **Basados en medidas de potencia de señal**. Ahora se mide la potencia de señal recibida desde los distintos puntos de emisión. En la figura 16 se puede observar el escenario para las medidas en función de la potencia de la señal.

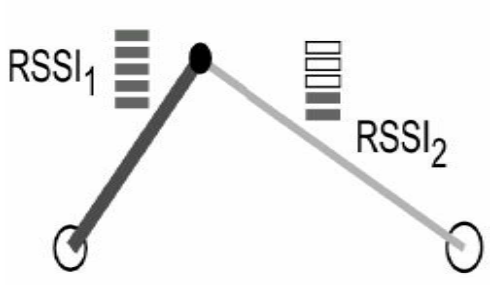


Figura 16. Intersección de señales basadas en medidas de potencia (obtenida de [11])

En la tabla 3 se presenta un resumen del observable y las situaciones para este tipo de tecnología

Observable	Situación
RSSI *(Received Signal Strength Indicator)*	Sistemas de comunicación de datos (WiFi, Bluetooth, GSM, etc) Sistemas basados en marcadores RFID No requieren línea de visión entre móvil y balizas

Tabla 3. Observable y situación para medidas basadas en la potencia de la señal (obtenida de [11])

- **Basados en medidas de transferencia de datos**. Medimos la calidad del enlace en función del número de errores que cometemos en una transmisión. Cuanto mayor distancia haya entre el punto de observación y el punto de emisión mayor será la tasa de error obtenida. Este sistema requiere realizar un entrenamiento. La figura 17 representa el escenario para la medida en función de la transferencia de datos.

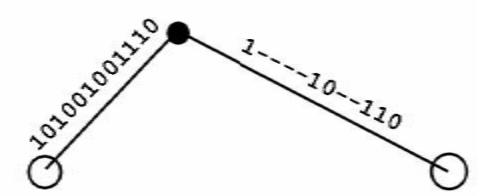


Figura 17. Medidas basadas en la transferencia datos (obtenida de [11]).

En la tabla 4 se presentan los observables y la situación en que se pueden realizar medidas para esta tecnología.

Observable	Situación
LQI (*Link Quality Indicator)* BER (*Bit Error Rate)*	Sistemas de comunicación de datos

Tabla 4. Observable y situación de medidas basadas en la transferencia de datos (obtenida de [11])

- **Basados en medidas de conexión**. Este método realiza su decisión en función de la celda a la que nos conectemos. En la figura 18 se presenta un posible escenario con tres celdas.

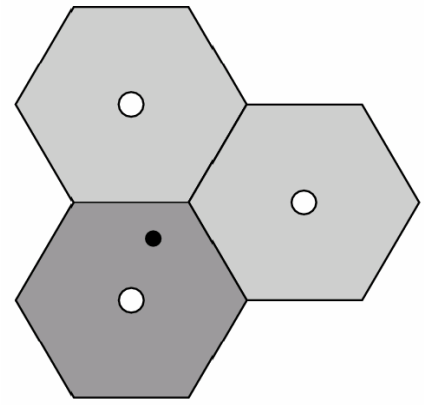


Figura 18. Escenario de medidas basadas en la celda de conexión (obtenida de [11])

c) *Según la privacidad del usuario.* Donde existen dos opciones:

- Sistemas centralizados, en los que el dispositivo móvil emite una señal de localización que los sensores del entorno detectan. Por ejemplo este es el sistema que se utiliza para la localización de la telefonía móvil por celda de conexión. Se muestra un ejemplo visual en la figura 19.

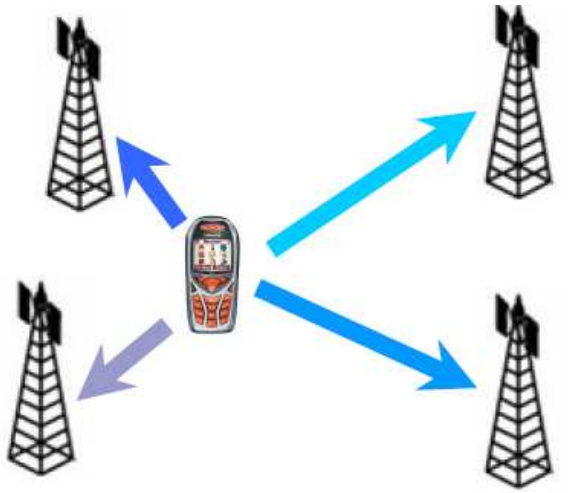


Figura 19. Escenarios de sistemas centralizados (obtenida de [11])

- Sistemas orientados a la privacidad, en los que el dispositivo móvil capta las señales de localización emitidas por los sensores del entorno. Adicionalmente puede existir un enlace de comunicación inalámbrica para comunicar los datos de posición. Se muestra un ejemplo visual en la figura 20.

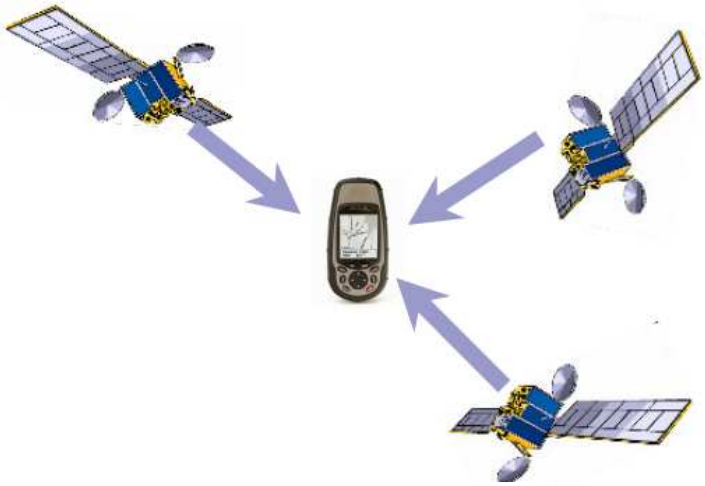


Figura 20. Escenario de sistemas orientados a la privacidad (obtenida de [11])

d) *Según se trate de localizar a un usuario individual o colectivo.* Si el nodo móvil tiene acceso directo a un número suficiente de estaciones con posición conocida, puede estimar su posición de forma individual. Este método de localización se denomina one-*hop*. Se puede observar un ejemplo gráfico en la figura 21.

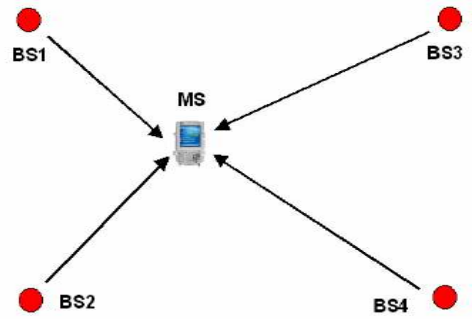


Figura 21. Localización One-*Hop* (obtenida de [11])

Si el nodo móvil sólo dispone de acceso a las estaciones base a través de otros nodos móviles, toda la red debe estimar su posición de forma cooperativa. Su nombre es localización *multi-hop*. En la figura 22 se puede ver un ejemplo de la localización multi-*hop*.

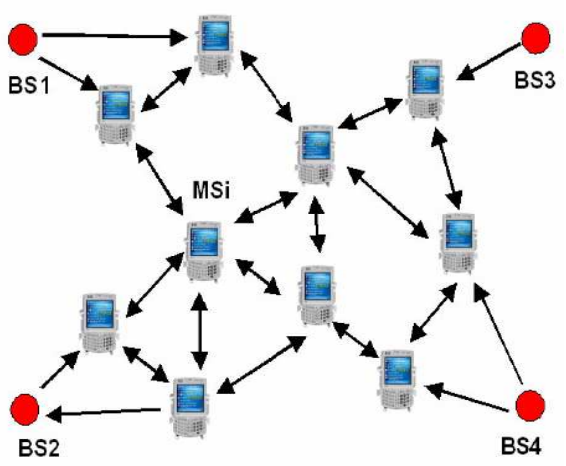


Figura 22. Localización Multi-*Hop* (obtenida de [11])

e) *Según cuánta precisión se necesite.* En este caso se usa métodos estadísticos con gran coste computacional para estimar la posición.

f) *Según qué método matemático de estimación se usa.* En el siguiente diagrama se puede observar el método de solución matemática utilizado en función de la

localización que se realice. Se presenta un cuadro resumen de la clasificación según el método matemático en la figura 23.

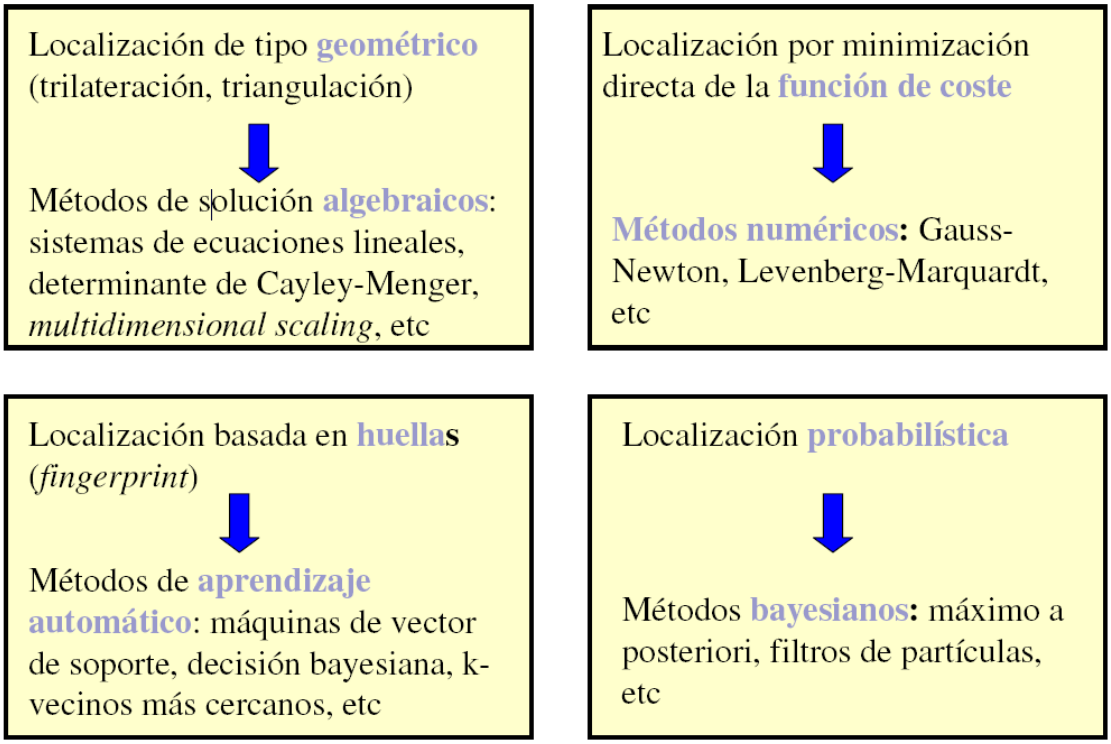


Figura 23. Cuadro resumen los distintos métodos matemáticos (obtenida de [11])

Por tanto, las características generales que definen a un sistema LPS basado en RF, según el **Máster Oficial en Sistemas Electrónicos Avanzados. Sistemas Inteligentes** son:

1. En general son capaces de operar con o sin visión directa entre móvil y balizas emisoras (LOS o NLOS).

2. La influencia del entorno es considerable (multicamino, atenuación causada por obstáculos, personas, etc).

3. Asimismo, resultan afectados por interferencias provenientes de otros sistemas de RF, o entre ellos mismos (MAI).

4. La predicción determinista del comportamiento de la señal de RF es imposible en condiciones realistas.

5. Los sistemas RF-LPS pueden beneficiarse de infraestructuras ya presentes

6. La diversidad espacial obtenida por combinar medidas provenientes de diferentes señales RF aumenta la precisión (distinta tecnología o distintos canales).

2.2.3 Tecnologías RF usadas en localización.

Sistema de posicionamiento Global (GPS).

Se puede argumentar que el sistema GPS ha resuelto el problema de localización en exteriores. Se trata de una tecnología matriz de nuevas aplicaciones, muchas de ellas insospechadas en el momento de su creación.

El sistema GPS se basa en la medida de señales codificadas emitidas desde una constelación de satélites que se encuentran orbitando alrededor de la tierra. El receptor recibe la información de la posición de manera precisa en el propio mensaje que envían los satélites. Para lograr esta precisión los satélites están provistos de relojes atómicos. Para conocer la posición se realiza una triangulación hiperbólica de las señales recibidas de los distintos satélites. Para conseguir mejorar la precisión con un factor de entre 5 y 10 se usan estaciones en tierra. Sin embargo, esta tecnología no sirve en interiores debido a su baja sensibilidad.

Radio de banda ultra ancha (UWB).

UWB es una tecnología en el rango de las PAN que permite el envío de paquetes de información en distancias cortas, de unos pocos metros.

Esta tecnología tiene origen militar, pero se empieza a usar en el ámbito civil desde 1990. Usa un gran ancho de banda del espectro de RF para transmitir información, por lo que es capaz de transmitir más información en menos tiempo que las otras tecnologías.

Una de las ventajas de este sistema es que hace uso de un espectro de frecuencia recientemente legalizado. Los sistemas UWB pueden usar frecuencias que van desde 3.1 GHz hasta 10.6 GHz: una banda de más de 7 GHz de anchura. Gracias a este gran ancho de banda cada canal de radio tiene una anchura de más de 500 MHz. Esta banda es mayor que las que usan otros sistemas que tienen un funcionamiento similar. Por ejemplo los canales que necesita la televisión son de 4,5MHz (*Sistemas de comunicacio*n*es Electrónicas*, Wayne Tomasi).

Sin embargo, el hecho de estar compartiendo bandas de frecuencia con otros dispositivos provoca que éstos, que se están comunicando, tengan que estar relativamente cerca para evitar interferencias.

Otras ventajas que ofrece la tecnología UWB son su bajo consumo, su bajo coste y su alta productividad, lo que marca esta tecnología como el futuro de las WPAN.

Además, permite la reutilización de espectros. Por ejemplo, se puede tener una serie de dispositivos comunicándose con nuestro ordenador a través de un canal en una habitación y a la vez, en otra habitación, otra serie de dispositivos en el mismo canal comunicándose igualmente.

En localización en interiores, UWB tiene dos grandes ventajas:

- Casi siempre es posible identificar el componente con LOS (transmisión directa) y aproximar así el rango auténtico de emisor y detector.

- Su elevado ancho de banda (BW) permite una gran resolución en la medida de los retardos (~ 1 ns).

Las desventajas son que requiere sincronización precisa entre lectores, que la detección de llegada de pulsos se realiza mediante filtros adaptados y que la estimación se basa en la medida de TOA (AOA es inservible por el multicamino).

En el siguiente gráfico, que se muestra en la figura 24, se puede ver una comparativa visual de las tecnologías con distintos anchos de banda. Además, se puede apreciar que cuanto mayor sea el ancho de banda menor densidad espectral necesitará el sistema.

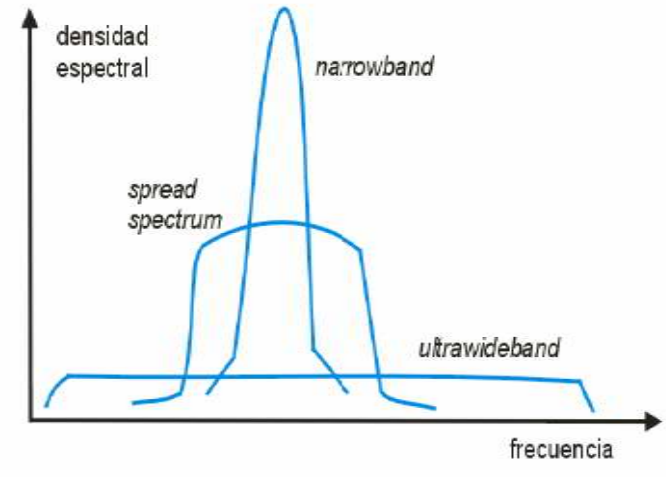


Figura 24. Comparativa de los distintos anchos de banda (obtenido de [11])

La tecnología Wi-Fi/Bluetooth/Wimax.

Se trata de estándares que tienen ventajas inmediatas como la estandarización, su ubicuidad y su bajo coste. Han sido diseñadas para conseguir alta velocidad de transmisión y no en particular para minimizar el consumo de energía o el coste de los nodos.

No están diseñadas para medir TOF, por lo que la mayoría de aparatos sólo pueden usar RSSI. Entre las tres tecnologías abarcan rangos de 1 m hasta varios km, con precisiones típicas de 1-5 m, dependiendo de:

(a) la densidad de emisores de RF del área

(b) la complejidad del entorno

(c) el tiempo de promediado

En la tabla 5 se puede ver un resumen de estas tres tecnologías:

Nombre	Frecuencia	Alcance	Características
Bluetooth 802.15	2.4 GHz	1 m (clase 3: 1 mW) 10 m (clase 2: 2.5 mW) 100 m (clase 1: 100 mW)	Redes de pequeño tamaño (piconets) diseñadas para PAN (∼USB inalámbrico). Fácil conexión de elementos sin mucha configuración. Los estándares futuros prevén UWB.
Wifi 802.11	2.4 GHz 5 GHz	P=100 mW 100 m (ext) ∼ 30 m (int)	Redes completas diseñadas para LAN (∼Ethernet inalámbrico). Requiere configuración de nodos. Completamente estandarizado y bajo coste de los chipsets.
WiMAX 802.16	2.3 GHz 2.5 GHz 3.5 GHz	< 50 km	Diseñada para MAN (∼ADSL inalámbrico) y cubrir el "último km". Posiblemente converja con 4G (conectividad "nómada"). Tecnología aún en desarrollo.

Tabla 5. Resumen de las tecnologías Bluetooth, Wi-Fi y Wimax

Zigbee

ZigBee es el nombre de la especificación de un conjunto de protocolos de alto nivel de comunicación inalámbrica para su utilización con radios digitales de bajo consumo, basada en el estándar IEEE 802.15.4 de redes inalámbricas de área personal (*wireless personal area network*, WPAN). Su objetivo son las aplicaciones que requieren comunicaciones seguras con baja tasa de envío de datos y maximización de la vida útil de sus baterías.

En principio, el ámbito donde se prevé que esta tecnología cobre más fuerza es en domótica. La razón de ello son diversas características que lo diferencian de otras tecnologías:

- Su bajo consumo
- Su topología de red en malla
- Su fácil integración (se pueden fabricar nodos con muy poca electrónica).

En la siguiente figura, la figura 25, se puede observar un ejemplo de una habitación domótica, que usa tecnología ZigBee.

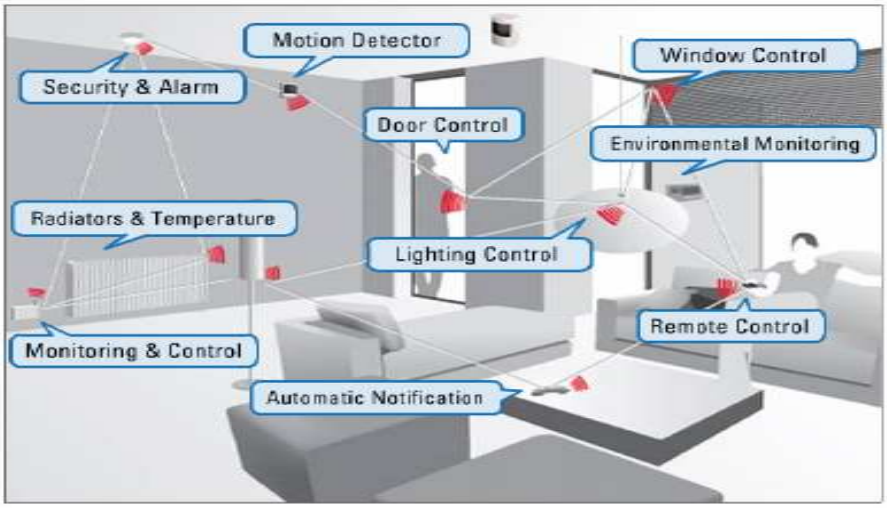


Figura 25. Ejemplo de casa domótica (obtenida de [11])

Marcadores de radiofrecuencia (RFID)

RFID (*Radiofrequency identification*) es un sistema de identificación basado en la transmisión de una señal de RF desde un emisor (marcador o *tag*) a un lector.

Los marcadores pueden ser activos (con baterías) o pasivos (usando la energía proporcionada por la señal de interrogación). Rangos de medida: **Activos**=10-100 m; **Pasivos** < 1 m.

Un lector de RFID recibe los códigos de identificación (*Tag* ID) de marcadores colocados en posiciones conocidas del entorno y usa la fuerza de señal recibida de cada uno (RSSI) para estimar su posición.

Ha tenido lugar una implantación masiva en los últimos años de esta tecnología para actividades tan diversas como:

- Control de inventario en tiendas
- Identificación de ganado y mascotas
- Peajes
- Control de acceso a edificios
- Pasaportes “digitales”

2.3 Posicionamiento en interior mediante Wi-Fi.

2.3.1 Introducción

Una WLAN es un sistema de comunicaciones de datos que transmite y recibe datos utilizando ondas electromagnéticas, en lugar del par trenzado, coaxial o fibra óptica utilizados en las LAN convencionales, proporcionando conectividad inalámbrica dentro de un área de cobertura.

Las WLAN se encuadran dentro de los estándares desarrollados por el IEEE para redes locales inalámbricas y cumplen con los estándares genéricos aplicables al mundo de las LAN cableadas (IEEE 802.3 o estándares equivalentes), pero necesitan una normativa específica adicional que defina el uso y acceso de los recursos radioeléctricos. Estas normativas definen de forma detallada los protocolos de la capa física (PHY), la capa de Control de Acceso al Medio (MAC) y Control del Enlace de Datos (DLC) que regulan la conexión. El primer estándar de WLAN lo generó el organismo IEEE en 1997 y se denomina IEEE 802.11. Se puede observar un esquema de las redes inalámbricas en la figura 26.

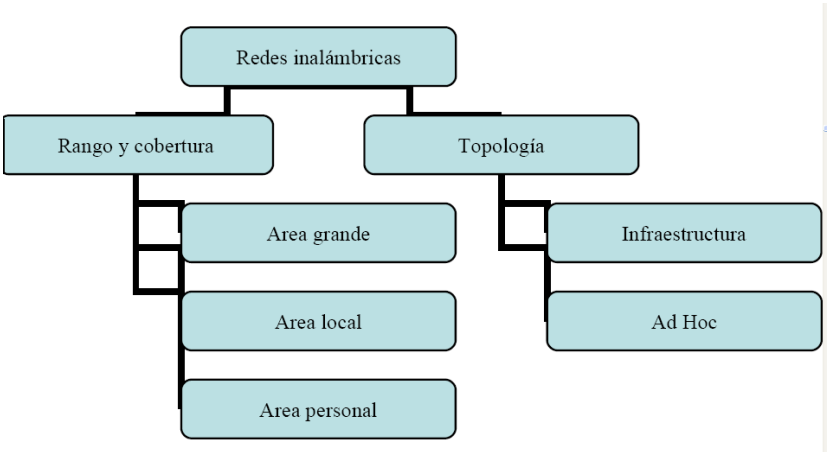


Figura 26. Esquema de las redes inalámbricas (obtenida de [11])

Desde entonces varios organismos internacionales han desarrollado y generado nuevos estándares.

2.3.2 Configuraciones o estándares

Hay tres configuraciones típicas:

- *Redes "ad-hoc".* Son redes cerradas sin infraestructura donde un grupo de terminales próximos se comunican entre sí.
- *Redes de acceso inalámbricas con infraestructura de red.* Se trata de redes donde los terminales se comunican con un punto de acceso a través del cual pueden acceder a redes externas.
- *Enlace entre varias WLAN o WMAN.* Se consigue interconectando LAN's o WLAN's distantes.

En la tabla 6 podemos ver un resumen de los distintos estándares de redes WLAN existentes actualmente.

Características

Estándar	Banda	Número de canales			Velocidad por canal		Alcance	Calidad de servicio
		Europa	USA	Asia	Máxima	efectiva		
802.11b	2,4 GHz	4	3	3	11 Mbit/s	6 Mbit/s	100 m	No
802.11a	5 GHz	0	12	4	54 Mbit/s	31 Mbit/s	80 m	No
802.11g	2,4 GHz	4	3	3	54 Mbit/s	12 Mbit/s	150 m	No
HiperLAN 2	5 GHz	15	12	4	54 Mbit/s	31 Mbit/s	80 m	Sí
5-UP	5 GHz	7	6	2	108 Mbit/ s	72 Mbit/s	80 m	Sí

Prestaciones de WLANs

[**A. Dornan. "Wireless LAN Standards". Network Magazine, vol. 17, nº 2**]

Tabla 6. Prestaciones de WLANs

También se pueden observar las características principales de cada uno de los elementos de la tabla anterior en la tabla 7.

Estándares

Origen	Estándar	Frec.	Nivel físico	Velocidad máxima	Disponible	Referencia
IEEE	802.11b	2,4 GHz	DSSS	11 Mbit/s	2001	standards.ieee.org/wireless www.wi-fi.org
IEEE	802.11a	5 GHz	OFDM	54 Mbit/s	2002	
IEEE	802.11g	2,4 GHz	OFDM/ DSSS	54 Mbit/s	Fin 2002	
ETSI	HiperLAN2	5 GHz	OFDM	54 Mbit/s	2003	www.hiperlan2.com www.etsi.org/bran
ETSI / IEEE	5GHz Unified Protocol (5-UP)	5 GHz	OFDM	108 Mbit/s	2003	
Bluetooth SIG	Bluetooth	2,4 GHz	DSSS / FHSS	0,721 Mbit/s	2002	www.bluetooth.com

Tabla 7. Estándares IEEE

IEEE 802.11a

El IEEE 802.11a permite alcanzar una velocidad máxima de 54 Mbit/s por canal, casi 5 veces superior a la velocidad máxima del 802.11b. Este estándar usa *Orthogonal Frequency Division Multiplexing* (OFDM – Multiplexación por división de frecuencia ortogonal), más complejo que *Direct Sequence Spread Spectrum* (DSSS) de 802.11b, por lo que ha tardado más en desarrollarse (a pesar de haberse definido antes como indica la letra 'a' de su nombre).

Este estándar usa frecuencias más altas, en la banda de 5 GHz, con espacio para más canales y menos pr*opens*a a interferencias que la banda de 2,4 GHz. El uso de frecuencias más altas produce una mayor absorción de las señales por obstáculos, lo que reduce el alcance de 802.11a comparado con el de 802.11b. Sin embargo, la robustez de OFDM frente a las interferencias multitrayecto compensa en parte la desventaja anterior, por lo que en la práctica el alcance de ambos sistemas es similar.

IEEE 802.11b

Conocido como "Wi-Fi" o como "*Wireless* Ethernet", define los niveles físicos y de acceso al medio (MAC). El acceso al medio se basa en un mecanismo de contienda similar al de Ethernet, sin calidad de servicio. Su velocidad es baja comparado con otras WLAN (aunque superior a la de las redes celulares 2G/3G) y su seguridad es bastante débil.

IEEE 802.11g

La versión IEEE 802.11g puede alcanzar la misma velocidad máxima que 802.11a pero usando la misma banda de 2,4 GHz que 802.11b con dos opciones: OFDM o DSSS mejorado. Es compatible con 802.11b pero con mayor alcance. Un mayor alcance permite cubrir una misma zona con menos puntos de acceso.

El problema es que debido a la mayor interferencia en la banda de 2,4 GHz, su velocidad efectiva tiende a ser menor. Por otra parte, el mayor alcance teórico de 802.11g no es aprovechable cuando la densidad de usuarios o el tráfico generado son altos o cuando hay problemas de interferencia entre redes próximas.

2.3.2.1 Presente de la tecnología

Las tres versiones 802.11a, b y g tienen como problemas comunes una débil seguridad, carencia de calidad de servicio y limitaciones de movilidad.

Los grupos IEEE 802.11i y 802.11x están trabajando en varias soluciones que mejoren la seguridad del cifrado y añadan funciones de autenticación. Se estudian tanto soluciones compatibles con WEP, como soluciones más robustas basadas en algoritmos nuevos, como el *Advanced Encryption Standard* (AES).

Por su parte, el grupo 802.11e pretende introducir calidad de servicio en 802.11b cambiando el mecanismo de acceso al medio basado en contienda por otro más controlado basado en TDMA.

El grupo 802.11f se ocupa de definir como realizar traspasos normalizados que permitan a un usuario cambiar de canal radio a otro o de un punto de acceso a otro sin perder la comunicación.

Otros grupos y actividades en curso son: el 802.11d, que está definiendo versiones de 802.11b en otras bandas de frecuencia; el 802.11h, que se ocupa de mejorar el control de potencia transmitida y la selección de canal radio del 802.11a en línea con los requisitos de ETSI; y el 802.11j, el más reciente, ocupado de la coexistencia de 802.11a con el estándar europeo HiperLAN2.

A la espera de que estas extensiones se normalicen definitivamente, algunos fabricantes ofrecen en sus productos soluciones propietarias para mejorar la seguridad o la movilidad entre canales y puntos de acceso.

2.3.2.2 Dispositivos.

Existen varios tipos de *hardware* que se pueden utilizar para conseguir una red inalámbrica Wi-Fi:

- **Adaptadores inalámbricos** o controladores de la interfaz de red (*wireless adaptaters* o *network interface controller* - NIC). Son tarjetas de red que cumplen con el estándar 802.11 y que permiten a un equipo conectarse a una red inalámbrica. Los adaptadores inalámbricos están disponibles en diversos formatos: tarjetas PCI, tarjetas PCMCIA y adaptadores USB.

- **Puntos de acceso** (denominadas zonas locales *de cobertura*). Permiten a los equipos equipados con tarjetas de red Wi-Fi acceder a una red. El punto de acceso es la unión entre la red y el usuario.

2.3.2.3 Modo de infraestructura

Cada estación informática se conecta a un punto de acceso a través de un adaptador inalámbrico.

La configuración formada por el punto de acceso y las estaciones ubicadas dentro del área de cobertura se llama *conjunto de servicio básico*. Cada BSS se identifica a través de un BSSID (identificador de BSS) que es un identificador de 6 bytes (suele corresponder al punto de acceso de la dirección MAC). En la figura 27 se presenta un escenario del modo infraestructura de una red Wi-Fi.

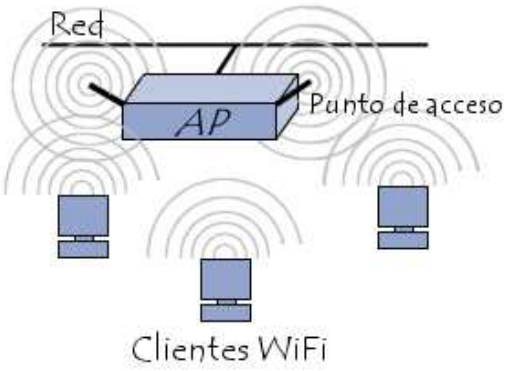


Figura 27. Escenario del modo infraestructura de una red Wi-Fi (obtenida de [11])

Es posible vincular varios puntos de acceso junto a varios BSS con una conexión llamada *sistema de distribución.* Entonces se pasa a tener lo que se conoce como un

conjunto de servicio extendido o *ESS*. El sistema de distribución también puede ser una red conectada, un cable entre dos puntos de acceso o una red inalámbrica.

Un ESS se identifica a través de un ESSID, que es un identificador de 32 caracteres en formato ASCII que actúa como su nombre en la red.

Cuando un usuario va desde un BSS a otro mientras se mueve dentro del ESS, el adaptador de la red inalámbrica de su equipo puede cambiarse de punto de acceso, según la calidad de la señal que reciba desde distintos puntos de acceso.

Los puntos de acceso se comunican entre sí a través de un sistema de distribución con el fin de intercambiar información sobre las estaciones incluso para transmitir datos desde estaciones móviles. Esta característica que permite a las estaciones moverse de un punto de acceso al otro se le conoce como itinerancia.

2.3.2.4 Modo ad hoc

Los equipos inalámbricos se conectan entre sí para formar una red punto a punto, es decir, una red en la que cada equipo actúa como cliente y como punto de acceso simultáneamente.

A la configuración que forman las estaciones se llama conjunto de servicio básico independiente o IBSS y es una red inalámbrica que tiene al menos dos estaciones y no usa ningún punto de acceso. Por eso, el IBSS crea una red temporal que le permite a la gente que esté en la misma sala intercambiar datos. Se identifica a través de un SSID de la misma manera en que lo hace un ESS en el modo infraestructura. En la figura 28 se presenta un escenario del modo ad hoc, sin infraestructura.

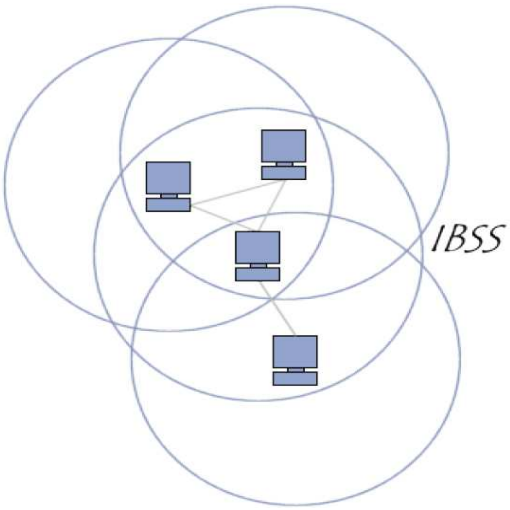


Figura 28. Escenario del modo ad hoc (obtenida de [11]).

En las redes ad hoc el rango del *BSS independiente* está determinado por el rango de cada estación. Es decir, que si dos estaciones de la red están fuera del rango de la otra, no podrán comunicarse, ni siquiera cuando puedan "ver" otras estaciones.

A diferencia del modo infraestructura, no tiene un sistema de distribución que pueda enviar tramas de datos desde una estación a la otra. Por lo tanto un IBSS es una red inalámbrica restringida.

2.3.2.5 Comunicación.

En una red Ethernet local común los equipos utilizan el método de acceso CSMA/CD, Por el que cada equipo tiene la libertad de comunicarse en cualquier momento. Cada equipo que envía un mensaje verifica que no haya otro equipo enviando un mensaje al mismo tiempo. Si alguno lo está haciendo, entonces ambos equipos deben esperar un período de tiempo aleatorio antes de comenzar a enviar el mensaje de nuevo.

Con la tecnología inalámbrica este proceso no es posible ya que dos estaciones que se comunican con un receptor no pueden escucharse entre sí al mismo tiempo debido a sus diferentes rangos de transmisión. Por esta razón, el estándar 802.11 utiliza un protocolo similar llamado CSMA/CA, donde se utiliza un mecanismo de evasión de colisiones basado en mensajes recíprocos de acuse de recibo que el transmisor y receptor intercambian, como se observa en la figura 29:

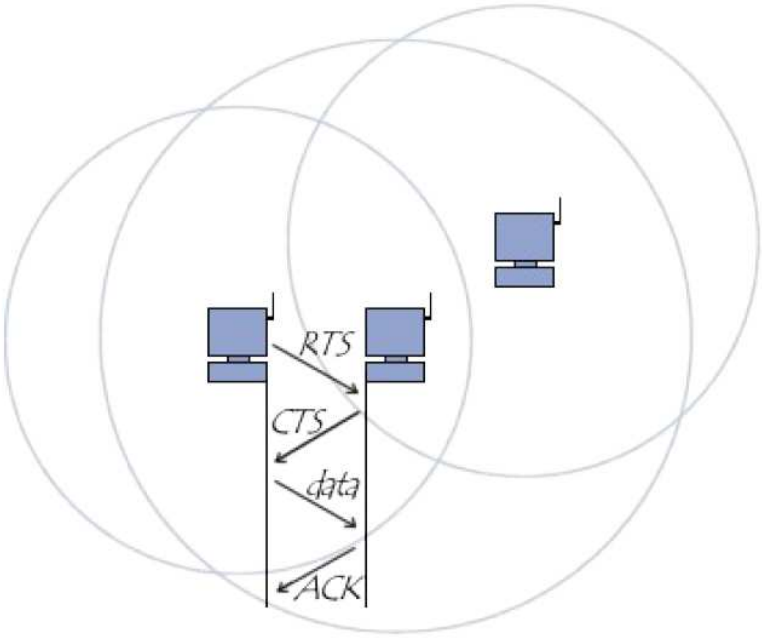


Figura 29. Mecanismo de evasión de colisiones (obtenida de [11]).

La estación que desea transmitir escucha a la red, si ésta está ocupada la transmisión se suspende hasta más tarde. Por el contrario, si el medio permanece libre durante un cierto período de tiempo (llamado DIFS, que es el espacio entre tramas), la estación puede transmitir la señal.

La estación transmite un mensaje "Listo para enviar" (o abreviado RTS, por "*Ready To Send*") con información sobre la cantidad de datos que desea enviar y su velocidad de transmisión.

El receptor, que por lo general es un punto de acceso, responde con un mensaje "Permitido para transmitir" (CTS por "*Clear To Send*") y después la estación comienza a enviar datos.

Cuando se han recibido todos los datos enviados por la estación, el receptor envía un aviso de acuse de recibo (ACK). En ese momento todas las estaciones cercanas esperan el tiempo estimado necesario para transmitir esa cantidad de información a la velocidad declarada.

2.3.2.6 Detección de Errores

El protocolo 802.11 en su capa MAC permite verificar la integridad de los datos enviados. Esto marca una diferencia fundamental con el estándar Ethernet, ya que éste no detecta errores.

La detección de errores se ha incluido en el nivel de enlace de datos debido a que en las redes inalámbricas el índice es mayor.

El índice de errores de transmisiones en redes inalámbricas se incrementa a menudo con paquetes de gran tamaño y por esta razón, el estándar 802.11 tiene un mecanismo de fragmentación que posibilita dividir una trama en varias partes llamadas "fragmentos".

El estándar 802.11 define el formato de las tramas de datos enviadas a través del uso del protocolo. Cada trama de datos está compuesta por un encabezado de 30 bytes (llamado encabezado MAC), un cuerpo y una FCS (Secuencia de verificación de trama) que permite corregir errores.

2.3.2.7 Interferencia y Atenuación

Debido a la naturaleza de la tecnología de radio, las señales de radio frecuencia pueden desvanecerse o bloquearse por materiales medioambientales. En función del tipo de material estas interferencias y atenuaciones varían.

- BAJA. Madera; Tabiques; Vidrio; Ventanas; Amianto; Techos; Yeso; Paredes interiores
- MEDIA. Ladrillo; Paredes interiores y exteriores; Hojas árboles y plantas
- ALTA. Agua; Lluvia; Niebla; Cerámica; Tejas; Papel; Vidrio; Plomo; Ventanas
- MUY ALTA. Metales; Vigas; Armarios

Otra fuente de interferencias son las redes inalámbricas que operan en el mismo espectro de frecuencias, afectando negativamente al rendimiento de esta tecnología.

Las tecnologías que pueden producir interferencias son las siguientes:

- Bluetooth
- Hornos Microondas
- Algunos teléfonos DECT inalámbricos
- Otras redes WLAN

2.3.2.8 Cálculo de la posición

La localización mediante redes *Wireless* puede llevarse a cabo de diferentes maneras:

- Vector de potencias.
- Triangulación de potencias.
- Métodos heurísticos.

Vector de Potencia

La información de la señal que emiten los puntos de acceso (AP) y que se recoge en los entrenamientos de los dispositivos, se almacena en una base de datos. Esta base de datos realmente es un vector donde cada celda contiene la potencia que le llega al usuario de cada AP a su posición en un instante determinado.

Este método se divide en tres fases:

1. En la primera fase hay que conocer la información de los APs.

2. En la segunda, la fase de entrenamiento, se construye la base de datos con el dispositivo, de forma que se guarden las potencias de cada AP para cada posición del mapa.

3. En la tercera fase, la fase de estimación, se obtiene para cada localización un vector con los niveles de señal recibida de cada uno de los APs. Este vector se compara con la base de datos para obtener la posición estimada como aquella en la que los niveles de señal son más cercanos o parecidos.

Triangulación de Potencia

Triangulación es un término cuyos orígenes son los círculos de la navegación, ya que se toman múltiples puntos de referencia para localizar una posición desconocida.
La triangulación se basa en las siguientes fases:

1. Con las potencias de tres puntos de acceso que llegan al cliente se crea un sistema de ecuaciones, que representa tres círculos.

2. Se resuelve el sistema de ecuaciones obteniendo un conjunto de puntos, llamados puntos de triangulación.

3. Cada punto de triangulación se considera el vértice de un triángulo.

4. Se forman todos los triángulos posibles y se calculan sus áreas para compararlas.

5. El centro del triángulo con el área más pequeña se toma como estimación de la ubicación del cliente

Heurísticas

Los métodos heurísticos se pueden utilizar por si solos, aunque su principal utilidad es la mejora de los dos métodos anteriores. Este método se basa en el descarte de posiciones de la base de datos para reducir las posibles localizaciones.

Para realizar este filtro de posiciones existen distintos criterios dentro de los métodos heurísticos:

- Heurística de Proximidad. Se basa en el punto de acceso más cercano al terminal para determinar su posición. Según la potencia que llega de cada

punto de acceso, se descartan los valores mínimos y se dice que la mayor potencia corresponde al del punto de acceso más cercano al dispositivo. Por lo tanto, se asume que el cliente está en la posición de dicho punto de acceso.

- Método de los vecinos más cercanos. El método de los *k*-vecinos más cercanos forma parte de una familia de técnicas de aprendizaje conocida como aprendizaje basado en ejemplos (*instance-based learning*).

  El aprendizaje en estos algoritmos consiste en memorizar los ejemplos de entrenamiento presentados. Esto quiere decir que cuando una nueva posición se le presenta al sistema de aprendizaje, un conjunto de ejemplos similares se recupera de la memoria para clasificar la nueva posición. Gracias a esto podemos conocer una ubicación aproximada del usuario.

  El inconveniente de esta técnica es que se requiere un gran número de puntos de calibración para poder realizar las comparaciones. Cuanto mayor sea el número de comparaciones mejor será el rendimiento de este método.

- Heurística de Movimiento. El movimiento es una parte importante del contexto de un usuario en un sistema de localización. En función de la fuerza de la señal recibida podemos clasificar al sujeto como parado o en movimiento. Para ello hay que tener en cuenta que la potencia señal de los APs tiene más picos alrededor de la posición estimada cuando el dispositivo está en el movimiento que cuando está parado.

  También hay que entrenar al dispositivo tanto en movimiento alrededor de una zona como parado en un punto de interés.

- Teoría de Bayes. Se trata de una técnica que mantiene una distribución de probabilidad sobre todas las posibles ubicaciones del entorno. Las técnicas probabilísticas consiguen una precisión superior que las técnicas deterministas, aunque a cambio tenemos un mayor coste computacional.

  El escenario óptimo para aplicar la aproximación Bayesiana es el de un entorno en forma de rejillas. Otra alternativa que modela el entorno es un mapa topológico. En este caso la localización se basa en el hecho de que el dispositivo identifica automáticamente que ha alcanzado un nodo del mapa en base a alguna información geométrica del entorno.

- Redes Neurales o Neuronales. Esta tecnología puede construir sistemas capaces de aprender, de adaptarse a condiciones variantes, o incluso si se dispone de

una colección suficiente de datos, predecir el estado futuro de algunos modelos. El entrenamiento en este caso se utiliza para agilizar el aprendizaje. Se memorizan las características de los puntos de interés al dispositivo y así es capaz de reconocer el área de localización en tiempo real.

2.4 Sistemas de posicionamiento basados en Wi-Fi.

2.4.1 Ekahau

Ekahau es el único *software* que hay en el mercado basado en tiempo real para sistemas de localización. Se basa en la calibración de la fuerza de señal y las huellas digitales que emiten las radiobalizas de la red *Wireless*. Funciona tanto en interior como en exterior siempre que haya cobertura *Wireless*.

Se trata de un sistema que es compatible tanto con los dispositivos pasivos de Ekahau: etiquetas o *Tags* T201, como con dispositivos activos: ordenadores portátiles, PDA's o cualquier elemento tecnológico con conectividad *Wireless*. Goza de una precisión de hasta 1 metro en las mejores condiciones de la red y del estado de los APs, lo que permite localizar muchos dispositivos a la vez y sobre el mismo mapa de situación, ya sean estos dispositivos activos o pasivos. La información de posición incluye las coordenadas x, y, el edificio, el piso, la habitación y la zona y funciona sobre cualquier estándar 802.11.

Para empezar a usar este sistema, además de la red *Wireless* necesitamos que el dispositivo del cliente tenga instalado el *software* Ekahau Client y un entrenamiento para calibrar los mapas del área de localización.

Ekahau utiliza el método del vector de potencia para calcular la posición del usuario a partir de la potencia de los APs. La fase de entrenamiento de los dispositivos se realiza midiendo las señales que se reciben de los APs en puntos de calibración y el cálculo de los vectores de posición se realiza de manera distinta según si el dispositivo es activo (PDA, ordenador) o pasivo (*tags*).

En la fase de estimación, para cada punto de la localización se compara la intensidad de señal que emiten los APs con la base de datos creada en la fase de entrenamiento y así determinar la posición del usuario.

Los elementos que conforman el sistema Ekahau son:

- EPE (Ekahau Positioning Engine)

Es el *software* de Ekahau que se utiliza como Centro de Control y es el que se encarga de hacer de plataforma de localización, es decir, desde donde se calcula y controla la posición de todos lo dispositivos clientes de Ekahau.

- Los *Access Points* en el área de localización.
  Permiten enviar la información de la red cableada hacia los clientes. En este sistema hay una condición para poder realizar la localización, como mínimo tres de los APs tienen que ser de nuestra red.

- Ekahau Client.
  Es el *software* que se debe instalar en el dispositivo del cliente (PDA, Tag, portátil…). Tiene que estar dotado de una tarjeta de red que incluya un transceptor radio y una antena, para poder ser localizado.

Para calcular la posición Ekahau se usa la técnica del Wi-Fi *Mapping* o Vector de Potencia además de una heurística de movimiento para mejorar la precisión del sistema. Por eso la fase de entrenamiento es la base para la localización del dispositivo. Para el cálculo de la posición en los dispositivos activos hay que instalar el *software* de Ekahau Client, ya que este *software* se encarga de medir las señales que provienen y se envían a los APs y crear el vector de potencias.

En el caso de las etiquetas o dispositivos pasivos son los APs los que calculan las coordenadas de la etiqueta en el área de localización y las envían al servidor. En la siguiente figura se puede ver el método de cálculo de la posición que utiliza Ekahau. El punto rojo es la posición del usuario que se estima a través de las potencias que le llegan de los APs que tiene a la vista.

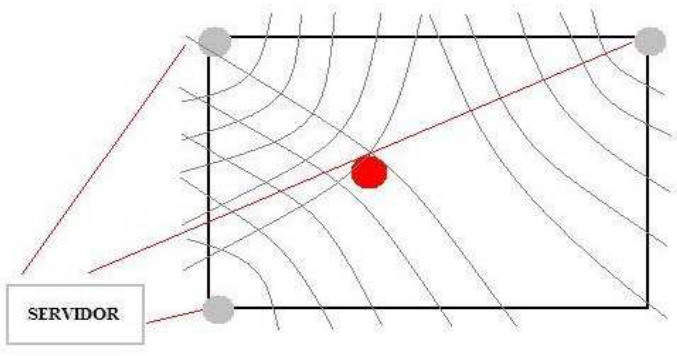


Figura 30. Escenario de un sistema Ekahau (obtenida de [10])

2.4.2 Place Lab

Place Lab es un *software* libre destinado a aplicaciones de cálculo de la posición de un usuario. Fue desarrollado por investigadores de la Universidad de Washington, de

California en Berkeley y por Intel en 2004. Está capacitado para obtener la localización del cliente tanto en interiores como en exteriores siempre que tenga cobertura *Wireless*. Al ser *software* libre permite el acceso y la modificación de los códigos fuente, escritos en el lenguaje Java, de forma que se pueda ajustar la aplicación según las funciones que se necesitan.

El sistema se caracteriza porque los clientes pueden determinar su ubicación sin necesidad de interactuar constantemente con un servicio central, es decir, el cálculo de la posición se realiza desde el propio dispositivo. Otra de las características es la privacidad de Place Lab ya que los dispositivos deben ser capaces de posicionarse basándose en la monitorización pasiva del entorno de localización.

Permite la localización de los dispositivos mediante Wi-Fi (balizas 802.11 o APs), torres GSM de teléfonos móviles, dispositivos fijos de Bluetooth y GPS. También es capaz de maximizar la cobertura en la mayoría de situaciones cotidianas, ya que reduce el tiempo y la dificultad de localización gracias a la gran cantidad de APs disponibles y su fácil mapeo.

La idea básica del cálculo de la posición es que todos los APs tienen un identificador único que se corresponde con la dirección de la capa física del protocolo 802.11 (dirección MAC). Este identificador reconoce los APs y sabe en qué posición están ubicados. Para determinar la posición del usuario el sistema ejecuta una aplicación que escucha los identificadores de las señales que llegan de los APs. Tras esto se busca la posición de los APs asociada a una zona del mapa y guardada en una base de datos. Para saber la posición del usuario se hace referir ésta a la localización de los APs. Es decir, el sistema se basa en la posición de los APs para calcular la posición del usuario. Los dispositivos que utilizan Place Lab sólo necesitan interactuar con los APs con el fin de adquirir sus identificadores.

Los elementos que conforman el sistema:

- Los *Access Points* en el ambiente: son imprescindibles para poder localizar un dispositivo mediante Place Lab. Los APs se reconocen mediante la dirección MAC, que los hace únicos. Mediante este identificador el cliente puede calcular su posición más fácilmente. En Place Lab, los APs no tienen porqué ser de nuestra red, se pueden utilizar cualquiera de los que tengamos a la vista.
- La base de datos: Place Lab depende de la localización de los APs para funcionar correctamente, por lo que necesita una base de datos que contenga las coordenadas de los APs y su dirección MAC. En nuestro caso hemos utilizado una base de datos local, que almacena la información de los APs en el ordenador que utilizamos como dispositivo cliente, pero si hay varios

dispositivos a localizar, se puede utilizar una base de datos remota a la que accedan todos los usuarios.

- Los Clientes de Place Lab: son los usuarios, que instalan el *software* en sus dispositivos. La funcionalidad del cliente se divide en tres partes: *Spotters*, Mappers y Trackers. Los *Spotters* son los responsables de observar el mundo físico y monitorizar la interfaz radio y compartir los identificadores de los APs con otros componentes del sistema. La función del Mapper es buscar en la base de datos la información de los APs conocidos, sus coordenadas, su dirección MAC y si son de tipo *Wireless*, Bluetooth o GPS. El Tracker es el componente del cliente que utiliza los datos proporcionados por el Spotter y el Mapper para estimar la posición del usuario. Los clientes de Place Lab no necesitan transmitir datos para determinar su posición.

La precisión media teórica de Place Lab es de entre 15 y 20 metros, o el doble del error que hay en GPS. La cobertura y la exactitud de Place Lab dependen del número y del tipo de AP que hay alrededor del dispositivo cliente.

Para calcular la posición en primer lugar hay que añadir en una base de datos que creamos nosotros, las coordenadas de los APs que forman nuestra red *Wireless* y su dirección MAC, para que puedan ser registrados como APs conocidos y el resto de APs que vea el dispositivo alrededor como desconocidos. Como se puede ver en la figura siguiente, los APs de nuestra red son reconocidos por la información que hemos introducido en la base de datos. El resto de APs se muestran como desconocidos.

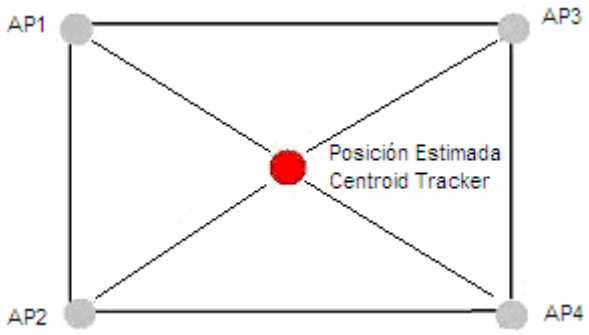


Figura 31. Escenario de un sistema Place Lab (obtenida de [12]).

Place Lab no tiene fase de entrenamiento, del reconocimiento de los APs ya se puede hacer la fase de estimación o localización de los dispositivos. En esta fase, según el Tracker que se utiliza, la heurística aplicada es diferente. Para una primera prueba o para comprobar el funcionamiento del sistema, se usa el CentroidTracker (basado en geometría), que a partir de las coordenadas de la posición de los APs calcula el punto medio de todas y considera que la posición del usuario es el punto calculado.

2.4.3 Herecast

Se trata de un sistema que proporciona servicios de posición basados en Wi-Fi. Usa una base de datos con la posición de todos los puntos de acceso, de forma que utiliza un algoritmo de referencia a la base de datos para el cálculo de la ubicación del usuario.

El sistema se caracteriza por:

- Privacidad: Los cálculos se hacen sobre el dispositivo, por lo que no se rastrea al usuario desde ningún servidor central.
- Utiliza un sistema de nombramiento simbólico y no de coordenadas, es decir, expresa su localización en términos que puedan entender los usuarios.
- Tiene una infraestructura abierta, accesible para todos los usuarios.
- Proporciona una resolución buena para poder actuar con un solo AP en una habitación grande.
- Para cada punto de acceso se almacena: país, provincia, ciudad, área, edificio (nombre y calle) y planta.
- La base de datos se va creando y modificando cuando un usuario descubre información de un nuevo punto de acceso. Esta información pueden verla todos los usuarios.

La idea básica del cálculo de la posición es que Herecast crea una base de datos donde se guarda la información para cada AP. Cada vez que nos acercamos a un AP desconocido se guardan sus datos (país, ciudad, provincia, edificio…) en la base de datos asociándole una posición. En el proceso de localización, al estar cerca de un AP conocido sabemos en qué posición estamos refiriéndola a la localización del AP.

Los elementos que conforman el sistema son:

- Los *Access Points*.
  Disponen de un código de identificación que, a nivel usuario, suele ser bastante inservible. Herecast relaciona estos códigos con cadenas de texto utilizadas para identificar la zona donde se encuentra el punto de acceso (proceso denominado *landmark*).
- La Base de datos.
  Herecast almacena toda la información referente al código del punto de acceso y la información de la zona. Cuando un usuario se conecta a un punto determinado, el programa busca el código en la base de datos y muestra en pantalla la información de la zona correspondiente. Las bases de datos las actualiza cualquier usuario (registrado) del mundo.

- *Software* Herecast
  En los dispositivos clientes hay que instalar el sistema para permitir la localización del aparato y diversas aplicaciones asociadas a ella.

La precisión de Herecast es exacta bajo un nivel de señal de un solo AP en un área grande o en varios espacios juntos. La posición que estima el sistema no son unas coordenadas concretas sino un área y unos datos que el usuario define en el entrenamiento y se guardan en la base de datos.

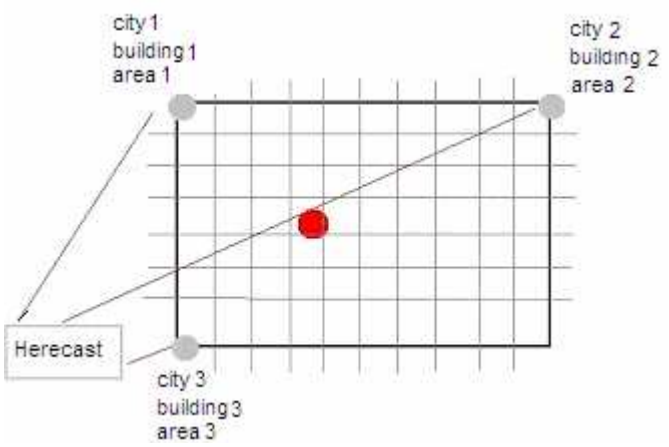


Figura 32. Escenario de un sistema Herecast (obtenida de [16]).

Para calcular la posición Herecast utiliza un sistema de nombramiento simbólico para reconocer un AP cercano, y mediante una heurística de proximidad, como método principal de localización, sabemos la posición en la que nos encontramos.

2.4.4 Comparativa teórica entre los tres sistemas.

Ekahau requiere que el usuario recoja el mismo los datos del área de localización, mientras que Place Lab incorpora un mapa radio y en Herecast los mismos usuarios son los recogen la información.

Como hay que recoger una gran cantidad de datos de calibración, Ekahau sólo es factible en ambientes pequeños. En Place Lab y Herecast los datos son contribuidos por muchos usuarios, por lo que la calibración es más reducida. El cliente de Ekahau necesita interactuar con el centro de control para determinar su posición, mientras que el cliente de Place Lab no requiere contactar con un servicio central.

Place Lab utiliza una heurística de proximidad o referencia a los AP, junto con el reconocimiento de los identificadores de los APs; mientras que Ekahau usa el método del Vector de Potencia y Herecast emplea un sistema de nombramiento simbólico y una heurística de cercanía.

Place Lab y Herecast son *software* libre al que podemos acceder y modificarlos para adaptarlos a nuestras necesidades. Ekahau es un producto comercial de una empresa que se dedica a experimentar y conocer la tecnología para el rastreo de la posición, por eso está probado en tantos campos de aplicación y es el más preciso en ambientes interiores.

Capítulo 3. Desarrollo del Sistema

3.1 Estudio de la viabilidad del sistema.

3.1.1 Alcance del proyecto.

Desde hace tiempo la localización en interiores viene siendo objeto de estudios e investigaciones. Hasta ahora no se ha conseguido obtener resultados tan satisfactorios como los obtenidos por los sistemas GPS. Las razones de esta diferencia son tanto técnicas, ya que la implementación es más complicada, como económicas porque se requieren un gran número de puntos de acceso, sensores, terminales, etcétera.

Como se ha visto en el capítulo anterior ya existen programas y sistemas que pueden servir para la localización en interiores. Sin embargo, con este proyecto se pretende ofrecer un sistema de posicionamiento completo para interiores, olvidándonos del exterior y tratando de completar la funcionalidad de los sistemas existentes.

El ámbito de este proyecto son las redes *Wireless* y el Sistema Operativo Android, por lo que el diseño se va a realizar teniendo en cuenta este entorno tan concreto. A pesar de ser un sistema muy concreto, lo que puede hacernos caer en el error de pensar que no se trata de una aplicación que pueda llegar a un gran número de usuarios, se trata de una aplicación totalmente abierta soportada por una tecnología emergente que pretende hacerse fuerte en el mercado de la tecnología portátil. Por ello, se puede afirmar que lo que se busca con este proyecto es realizar una aplicación fiable para un gran número de usuarios.

3.1.2 Funciones del programa.

3.1.2.1 Localización del usuario.

Para poder localizar al usuario, independientemente del tipo que sea, se va a necesitar un mapa de la planta donde éste se encuentre. Este mapa deberá ajustarse al tamaño de la pantalla del terminal y estar dividido en zonas. Para cada zona habrá un punto expresado en píxeles que será donde se dibuje en el mapa la posición del usuario.

Nuestra aplicación descargará el mapa y lo almacenará en la memoria externa del terminal. En este caso se ha decidido almacenar en el directorio raíz pero se puede guardar en la ruta que queramos.

Para localizar al usuario la aplicación realiza barridos de señal. El nivel de estas señales es comparado con los resultados obtenidos en el entrenamiento, y que están almacenados en el fichero del mapa de potencias. En el que caso de obtener una comparación positiva se pintaría en el mapa el punto en el que nos encontramos.

3.1.2.2 Generación del mapa de potencias.

El sistema necesita una aplicación capaz de diseñar el mapa de potencias, que luego permitirá el acceso a la información allí guardada para la estimación de la posición del usuario en el plano. Para el diseño de este mapa es necesario realizar un entrenamiento del sistema por parte del proveedor del servicio de localización. El cliente recibirá el mapa de potencias mediante una conexión a la red Wi-Fi.

El mapa de potencias se almacenará en el dispositivo del usuario que quiera saber su localización. El mapa de potencias será un fichero de texto que contendrá en cada línea el identificador de zona, el número MAC de cada punto de acceso detectado y su nivel de señal que hay en ese punto y las coordenadas expresadas en píxeles del mapa.

3.1.2.3 Actores en el sistema.

Existen dos tipos de actores en el sistema. Los primeros son los usuarios que desean conocer su posición en el mapa. Se trata de usuarios pasivos ya que la única acción que tienen que realizar es arrancar la aplicación.

El segundo tipo de usuario es activo. Es el encargado de la realización del mapa de potencias. Este usuario interactúa con el programa: le dice la posición en la que está y

le pide un barrido de frecuencias, para obtener la información necesaria para completar el mapa de potencias.

3.1.2.4 Descarga de archivos.

Para el funcionamiento óptimo del sistema cuando el usuario inicia el programa se produce la descarga de dos archivos.

El primer archivo es una imagen de la planta del edificio en el que el usuario se situará y que utilizará para pintar el punto que indicará la ubicación de éste.

El segundo se trata de un archivo de texto, que será el mapa de potencias. En este archivo se almacenarán los puntos de acceso y sus correspondientes niveles de potencia así como la posición en píxeles del mapa.

3.1.2.5 Características de la aplicación.

La aplicación debe tener un alto grado de manejabilidad, es decir, esta aplicación tiene que ser fácil de usar por cualquier tipo de usuario.

Como se ha dicho anteriormente, hay dos tipos de usuarios que interactúan con los mapas. Estos usuarios accederán a aplicaciones distintas, por lo que los usuarios que deseen conocer su posición en el mapa no tendrán acceso a la aplicación que permite calibrar el sistema.

3.1.2.6 Limitaciones.

El sistema que se trata de construir ha de funcionar en cualquier dispositivo que use como sistema operativo Android y con el *software* que se proporciona no se necesitará ni *hardware* ni *software* adicional.

Además, el dispositivo tendrá que disponer de una tarjeta de red capaz de detectar redes Wi-Fi, que utilizará para hacer la detección y extracción del nivel de potencia de las señales emitidas por los puntos de acceso.

Para una detección precisa será necesario realizar un estudio previo de la colocación de los puntos de acceso, para que no queden zonas muertas en las que no lleguen

señales, o zonas en las que sea difícil distinguir dos puntos con niveles de potencia similares.

3.1.3 Requisitos.

3.1.3.1 Requisitos generales

Requisito-01. Será necesario un mapa para poder mostrar la localización del usuario de forma gráfica.

Requisito-02. El mapa será un fichero con formato jpg, png o gif.

Requisito-03. El mapa será rectangular y no deberá salir del foco de la pantalla del dispositivo.

Requisito-04. El mapa representará los distintos pasillos de que se componga la planta del edificio. Está exento de aparecer el mobiliario.

Requisito-05. Tendrá que haber un número suficiente de puntos de acceso para la detección de la señal.

Requisito-06. El mapa se dividirá en zonas.

Requisito-07. Habrá dos aplicaciones, una para realizar el mapa de potencias y otra para estimar la posición de los usuarios del sistema.

Requisito-08. En la aplicación generadora del mapa de potencias el número de zonas en las que se entrenará el sistema será decidido por usuario responsable de la instalación del sistema en la planta del edificio en función de las características de dicha planta.

Requisito-09. Podrá completarse el entrenamiento en distintas fases. Es decir, no será necesario completar el entrenamiento en un único barrido.

Requisito-10. La aplicación no realizará transiciones entre las plantas del mismo edificio. Cada planta tendrá su propio mapa de potencias y mapa físico.

Requisito-11. Cuando el cliente inicia la aplicación, se realizan estimaciones de la posición constante y periódicamente hasta que éste finalice la aplicación o la resetee el sistema.

Requisito-12. Las potencias de las señales se almacenan en dBm.

Requisito-13. El usuario solo podrá ser localizado en las zonas definidas en la aplicación de entrenamiento.

Requisito-14. La localización se estimará comprobando los niveles de potencia que se reciben de los barridos y comparándose con los datos almacenados en el mapa de potencias.

Requisito-15. El usuario se descargará de un servidor los mapas de potencias y físicos cada vez que entren en las distintas plantas del edificio.

Requisito-16. Cada vez que el usuario cierra la aplicación de localización se borran los dos archivos descargados al iniciar la aplicación.

3.1.3.2 Requisitos de interfaces externos.

Como interfaces externas se tendrá las de comunicación con los puntos de acceso, la de conexión a la red para realizar la descarga de los archivos necesarios para el funcionamiento de la aplicación y la desconexión de la red una vez finalizada la descarga de archivos.

3.1.3.3 Requisitos de rendimiento.

No existirá ningún límite de dispositivos haciendo uso del sistema. La descarga de archivos deberá realizarse lo más rápido posible para que la aplicación comience su actividad lo antes posible.

3.1.3.4 Requisitos tecnológicos.

Tanto el usuario encargado de realizar el entrenamiento del sistema como el usuario que desea conocer su posición deben disponer de un dispositivo móvil que disponga de:

- Sistema operativo Android v1.5 o superior.
- Tarjeta de red que soporte la tecnología Wi-Fi.
- Espacio mínimo en la tarjeta de memoria externa de 5 Mb.

3.1.4 Seguridad.

Actualmente la seguridad que ofrece la aplicación son los *firewall* de los *routers* y de la red del departamento de Telemática, WLIT y la seguridad de la red Wi-Fi-UC3M. El uso de la aplicación no supone un compromiso para la seguridad de estas redes puesto que no se están enviando datos desde el dispositivo a la red, simplemente se está obteniendo información de la misma.

Como opción para unas líneas futuras, cada vez que se conecte un usuario a la red de localización deberá rellenar un formulario con una serie de datos. Estos datos podrán usarse tanto para controlar el acceso de los usuarios como para obtener información para realizar estadísticas sobre el rendimiento de la aplicación, sobre las zonas más visitadas, etcétera.

No se ha realizado un desarrollo exhaustivo de este punto ya que no era el objeto del proyecto, pero si que se es consciente de que sería una buena línea futura a seguir.

3.1.5 Planteamiento del problema.

Como se ha mencionado anteriormente, el sistema GPS no es válido para el posicionamiento en interiores, por ello se han de diseñar sistemas de posicionamiento Wi-Fi que permitan la localización de un dispositivo dentro de edificios.

Una de las ventajas que posee esta tecnología es la simplicidad, enfocándose todo el sistema en el diseño de una aplicación capaz de manipular los datos obtenidos de una red Wi-Fi.

El diseño de este sistema se va a basar en la medida de la intensidad de la señal, en la que el encargado de calcular la posición del dispositivo es el dispositivo del usuario. Una de las características de este sistema es que el posicionamiento se realiza por regiones.

El entorno físico de desarrollo será en la primera planta del edificio Torres Quevedo, la zona del Departamento de Telemática, usando como puntos de acceso, los *routers* que forman la red WLIT.

La ubicación gráfica sería la de la figura 33:

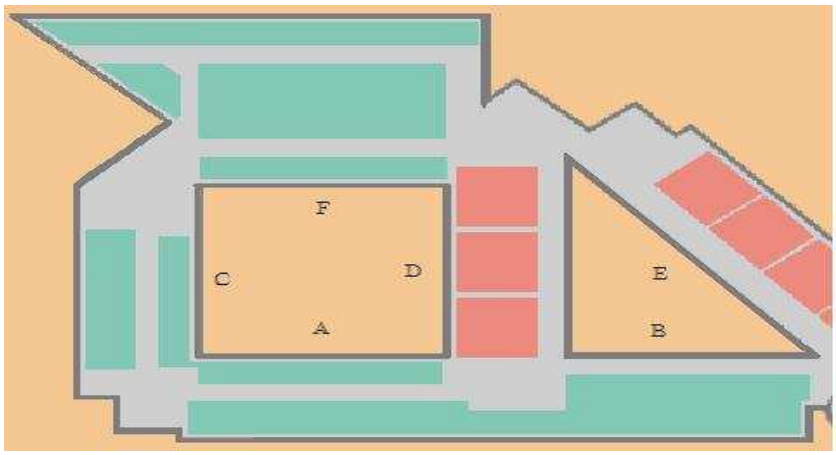


Figura 33. Plano de la planta del edificio Torres Quevedo.

Como la posición de los puntos de acceso es fija y conocida, cada punto de acceso informará al sistema de la intensidad de la señal emitida hasta el punto en que se encuentra el dispositivo receptor. A partir de esa medida se podrá realizar una interpretación de la posición comparando con un mapa de potencias, que previamente ha sido creado e instalado en el dispositivo.

Si se introdujese otro punto de acceso, se podrían hacer dos cosas. La primera es rehacer todos los cálculos con el nuevo punto de acceso y la segunda es completar el mapa de potencias con las nuevas zonas de cobertura, proporcionada por el nuevo punto de acceso.

Una de las ventajas del diseño del sistema, es que la posición del dispositivo cliente la calcula en el propio dispositivo, con lo cual se logra que el cálculo sea más rápido que haciendo consultas al servidor, como se demostrará en el capítulo de pruebas.

La definición de las regiones se es*table*ce arbitrariamente por los proveedores del servicio, en función de la distribución de la planta del edificio.

Los sectores de que se han decidido para este proyecto son los que se pueden ver en la figura 34:

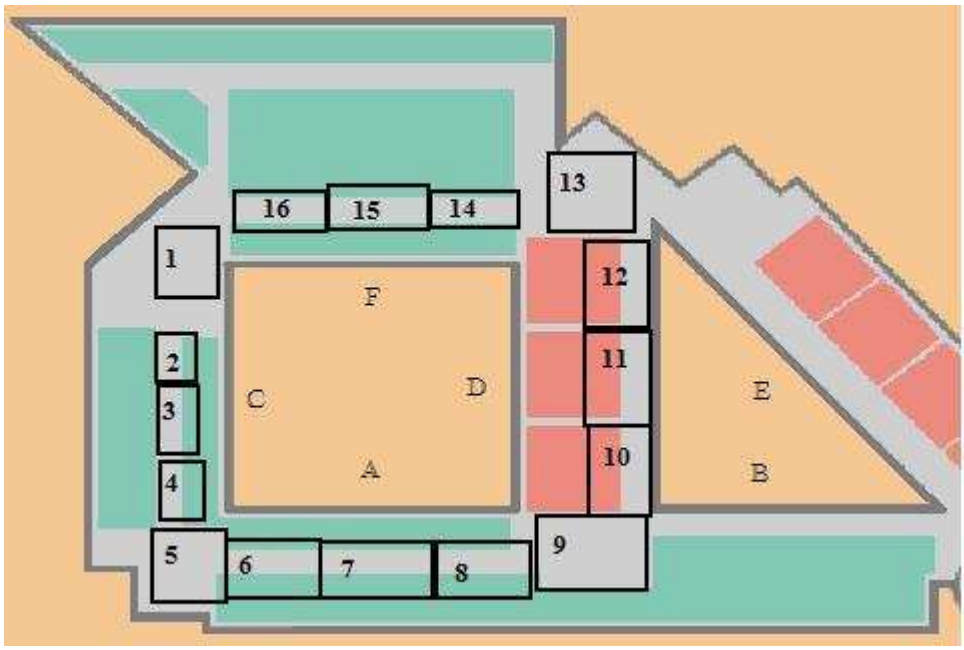


Figura 34. Sectores de localización.

3.2 Diseño del sistema.

3.2.1 Descripción general.

Este sistema de localización está compuesto por dos aplicaciones. La primera aplicación es la que genera el mapa de potencias, mientras que la segunda es la que obtiene la posición del usuario en el mapa y la representa.

Como se ha explicado anteriormente, no habrá un diálogo cliente servidor, sino que todos los datos necesarios para ejecutar las dos aplicaciones se conseguirán haciendo barrido de señales, guardando la información de cada uno y analizando los datos obtenidos.

Tan solo habrá una petición desde el cliente a un servidor web, que será para realizar la descarga de los planos al inicio de la aplicación.

3.2.2 Calibrador.

3.2.2.1 Contexto.

Se trata de la primera aplicación y está destinada a las personas que deseen montar la infraestructura del sistema de localización en la planta del edificio.

La aplicación se denominará `Calibrador` y en ella intervendrán dos actores, el dispositivo y la red de puntos de acceso Wi-Fi.

- Dispositivo móvil: su misión es la de crear el mapa de potencias para una nueva localización en la que se desee instalar el sistema de localización.

  Su tarea principal es la de realizar los barridos de señales para cada punto del mapa, que el usuario estime oportuno. Una vez realizada la acción y obtenidos los datos pertinentes se almacenará la información en un documento de texto.

  Para que la medida sea lo más fiel posible, se han realizado 250 barridos de señal para cada posición y después se ha aplicado la media aritmética en función del número de apariciones de cada punto de acceso. Con esto se consigue controlar las posibles variaciones momentáneas del nivel de potencia.

  En cada línea se guarda la información de un punto en concreto del mapa. Habrá tantas líneas como puntos desee el usuario encargado de completar el mapa de potencias. El formato de cada línea es el siguiente:

  <Zona X> <Mac 1> <Potencia1> <Mac2> <Potencia2>….<Píxel X> <Píxel Y>

También es capaz de completar el mapa con distintos barridos en momentos sucesivos, añadiendo nuevas líneas al fichero.

- Puntos de acceso: en este caso son actores pasivos, cuya misión es ceder información al dispositivo.

  La información que la aplicación requiere de cada punto de acceso es el número MAC. El dispositivo calcula la potencia de señal que recibe y almacena ambos datos.

  Los puntos de acceso están distribuidos por la planta y en función del lugar de localización del usuario se recibirá mayor o menor intensidad de señal. La distribución de estos puntos es importante para que la cobertura de localización sea lo mayor y más precisa posible.

3.2.2.2 Casos de uso.

Los casos de usos que tiene que cumplir la aplicación Calibrador son los que se pueden observar en la figura 35.

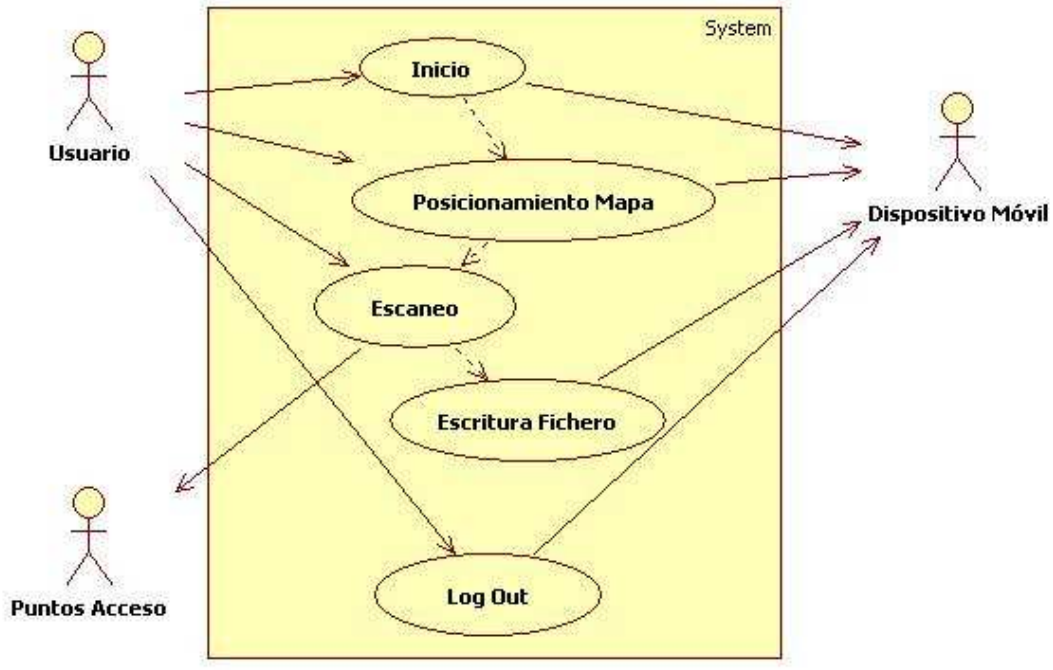


Figura 35. Diagrama de casos de uso de la aplicación Calibrador

- Caso de uso 1. Inicio.

  En este caso de uso se muestra una pantalla de bienvenida en la que se muestra información sobre la aplicación que se está ejecutando, la versión de la misma y cuanta información sea necesaria.

  Esta pantalla aparecerá durante unos segundos y dará paso a la siguiente ventana, que será el mapa de la planta que será objeto del estudio.

- Caso de uso 2. Posicionamiento en el mapa.

  Este caso de uso obtiene en píxeles las coordenadas del punto donde se encuentra el usuario encargado de generar el mapa de potencias.

  Para obtener estas coordenadas el sujeto debe pulsar sobre la pantalla en la posición en que se sitúa en el mapa. Una vez pulsado se pinta un punto en el mapa correspondiente a la zona que se ha pulsado.

  Se puede repetir la acción tantas veces como sea necesario. El sistema guardará la última pulsación.

- Caso de uso 3. Escaneo.

  Se encargará de realizar un escaneo de las señales que recibimos en el dispositivo móvil, filtrando las señales de la red que interesa para la localización. Se realiza un filtrado ya que se pueden detectar redes que no correspondan al sistema de localización que lo que harán será disminuir la agilidad de la aplicación.

  Cada escaneo se repetirá 250 veces y se hará la media, con el fin de obtener una medida fiel del nivel de señal en cada punto.

  Cada vez que se realice el escaneo y este finalice se mostrará un mensaje por la pantalla indicando al usuario, que el dispositivo está listo para realizar el siguiente barrido.

- Caso de uso 4. Escritura en fichero.

  Este caso de uso se encargará de crear un fichero de texto. En cada línea del fichero escribirá cada uno de los barridos realizados y cada punto de acceso de detectado, así como su nivel de potencia.

El fichero de texto se almacenará en la memoria del dispositivo y se irá completando cada vez que el usuario desee.

Si el fichero se ha creado anteriormente la aplicación completará dicho fichero con las nuevas posiciones que se le añadan.

- Caso de uso 5. *Log out*.
  Es el caso de uso encargado de cerrar la aplicación cuando el usuario considere oportuno hacerlo.

  Para cerrar la aplicación no tiene que haber lanzado ningún proceso. Es decir, el programa se tiene que encontrar a la espera de una petición de escaneo.

  Una vez se ha cerrado la aplicación el fichero del mapa de potencias queda guardado en la memoria, a disposición del usuario.

3.2.2.3 Diseño lógico.

Para analizar el diseño lógico se va presentar la estructura estática y los escenarios de ejecución de la aplicación.

3.2.2.3.1 Estructura estática del sistema.

La aplicación es la encargada del escaneo de redes y del almacenamiento de los datos para cada punto del mapa.

Con el objetivo de conseguir una implementación más sencilla se ha dividido la aplicación en tres bloques:

- Escáner. Es el encargado de resolver toda la casuística del barrido de redes y señales, además de presentar los datos de interés para la aplicación.

- Pintor. Este bloque se encarga de obtener la posición del mapa en píxeles y de pintar el punto en el mapa.

- Escritor. La función del bloque escritor será la de guardar los datos obtenidos por los dos anteriores bloques en un fichero de texto.

En las siguientes tablas se expondrá las responsabilidades de cada uno de los bloques anteriores.

Bloque	Escáner
Responsabilidades	Mediante este bloque se accederá a las redes Wi-Fi presentes en el edificio. Realiza el protocolo de comunicación con la red, obteniendo los datos necesarios para completar el mapa de potencias. Selecciona los datos útiles para esta aplicación, despreciando el resto y haciendo la aplicación más veloz.
Colaboradores	Escritor
Notas	No pueden realizarse dos operaciones Wi-Fi simultáneamente.
Incidencias	Si se conecta a una red Wi-Fi y a continuación se realiza un escaneo, la aplicación sufrirá un error y se cerrará.

Tabla 8. Bloque de Escáner

Bloque	Escritor
Responsabilidades	Proporciona toda la operativa necesaria para la escritura de los datos obtenidos por el sistema en un fichero de texto, el cual al finalizar el entrenamiento conformará el mapa de potencias.
Colaboradores	Pintor Escáner
Notas	Cada línea del fichero pertenece a una zona del mapa físico.
Incidencias	

Tabla 9. Bloque Escritor

Bloque	Pintor
Responsabilidades	Con este bloque se consigue manejar la parte gráfica del mapa, para pintar el punto en el que el usuario se encuentra y obtener esa posición transformada en píxeles.
Colaboradores	Escritor Escáner
Notas	Cada acción de pintar conlleva un refresco de la pantalla, no se carga un mapa nuevo.
Incidencias	

Tabla 10. Bloque Pintor.

3.2.2.3.2 Escenarios de la aplicación.

Inicio.

En este escenario se muestra la pantalla de inicial de la aplicación. Esta pantalla aparecerá durante unos segundos y dará paso a la pantalla en la que se desarrollará el trabajo principal de la aplicación.

Este escenario de inicio es en sí mismo una actividad, que cuando finaliza se queda en pausa esperando que el liberador de memoria interno lo elimine o que se vuelva a activar por una orden del usuario al manejar el dispositivo.

Este escenario responde al siguiente diagrama de estados que se presenta en la figura 36:

Figura 36. Diagrama de estados del caso de uso Inicio.

Posicionamiento en el *mapa.*

El primer objetivo de este escenario es obtener las coordenadas de un punto del mapa, expresadas en píxeles.

El segundo objetivo es el de pintar en las coordenadas anteriores un círculo, que indicará al sujeto que este manejando la aplicación el punto que ha seleccionado.

El funcionamiento es el siguiente: existe una clase que actúa cuando se ha producido una pulsación en la pantalla. Una vez se haya producido esta pulsación se realiza una consulta de las coordenadas de la pantalla. La aplicación guarda estas coordenadas, en píxeles, y llama al método correspondiente para que pinte el círculo en las coordenadas obtenidas anteriormente.

Una vez pintado el círculo la aplicación está a la espera de la siguiente pulsación sobre la pantalla.

El usuario puede repetir tantas veces sea necesario esta acción ya que la aplicación guardará los datos correspondientes al último toque en la pantalla.

En la figura 37 se muestra el diagrama de estados que representa lo anteriormente descrito:

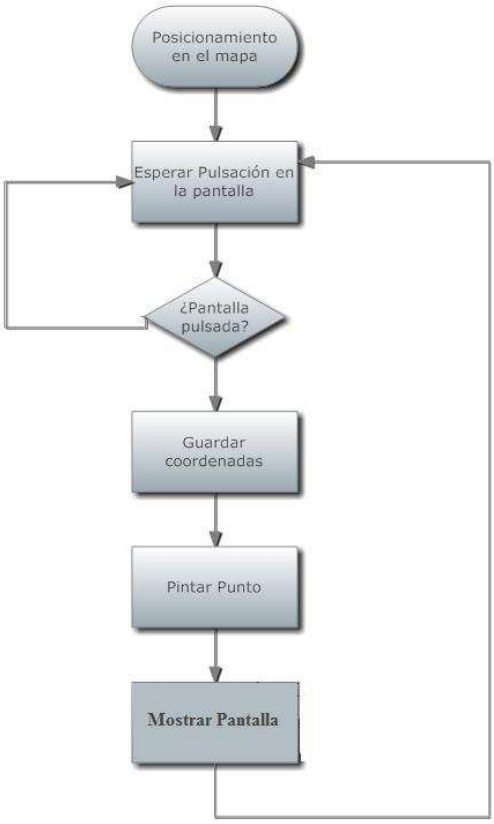


Figura 37. Diagrama de estados del caso de uso de Posicionamiento en el mapa

Escáner.

Este escenario contiene la funcionalidad principal de la aplicación y es el escenario que sirve como nexo de unión entre los distintos escenarios de la aplicación.

La aplicación está a la espera de que ocurra algo. Como se ha visto en escenarios anteriores, esto que espera la aplicación puede ser una pulsación en la pantalla, pero también puede ser una petición de escaneo.

Para realizar esta petición de escaneo, hay que entrar en el menú de la aplicación con el terminal y pulsar el botón correspondiente. Una vez realizada esta acción se procederá a realizar un barrido de las señales que el dispositivo detecta, este barrido se realizará 250 veces.

Una vez finalizado en el escáner, se mostrará por pantalla un mensaje indicando que la aplicación ha finalizado su escaneo de señales. En este punto el usuario decide realizar otro escáner o finalizar la aplicación.

Tras cada escáner se realiza la media de los datos obtenidos y se guardan los resultados en un fichero de texto que al finalizar la aplicación será el mapa de potencias completo.

En la figura 38 se muestra el diagrama de flujo corresponde al funcionamiento de este escenario:

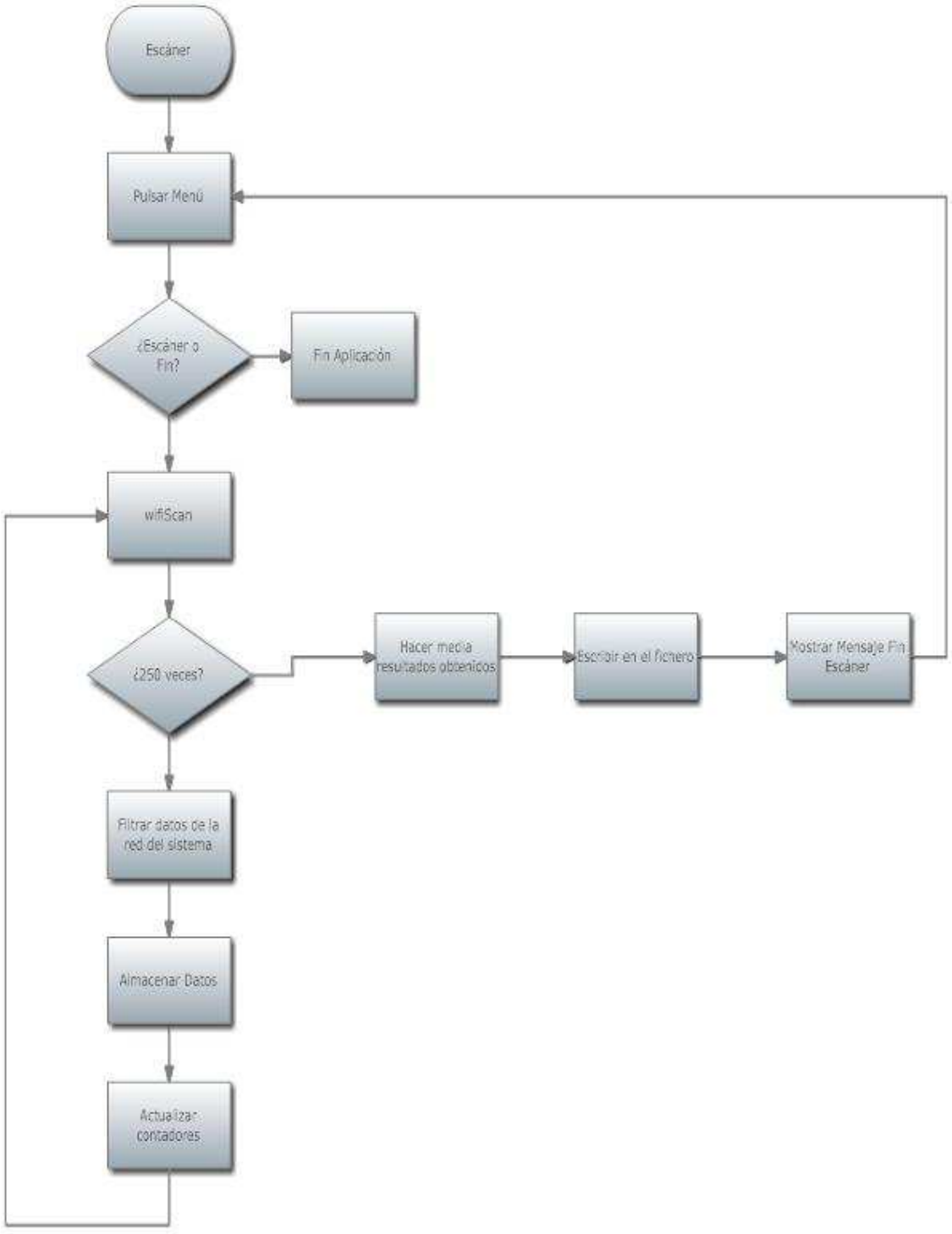


Figura 38. Diagrama de estados del caso de uso Escáner.

Escritura en el *fichero.*

Con este escenario se consigue que la aplicación guarde los datos obtenidos durante el escáner.

Cada escáner se guardará en una línea, pudiendo ejecutar la aplicación en distintos momentos, ya que la aplicación no sobrescribe el fichero, sino que lo actualiza añadiendo las nuevas líneas que el sujeto que maneja el dispositivo desea.

El formato de cada línea será el siguiente:

<Zona X> <Mac 1> <Potencia1> <Mac2> <Potencia2>….<Píxel X> <Píxel Y>

Si no existiese un fichero que se hubiese creado anteriormente, la aplicación lo crearía y comenzaría a escribir en él.

El siguiente diagrama representa el proceso de escritura:

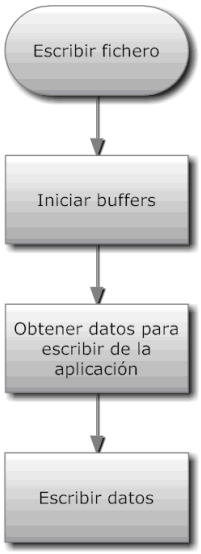


Figura 39. Diagrama de estados del caso de uso Escritura en el Fichero

Cierre de la aplicación.

Con este escenario se finaliza la aplicación. Para ello hay que pulsar sobre el menú de la aplicación y ahí pulsar el botón correspondiente. Se llamará al método `onDestroy()`, que realizará las operaciones internas del dispositivo para cerrar la aplicación.

Una vez realizada esta acción no se podrá volver hacia atrás a la aplicación, ya que se ha eliminado de la pila y no hay ninguna referencia es ésta. Para volver a usar la aplicación hay que ejecutarla desde el principio.

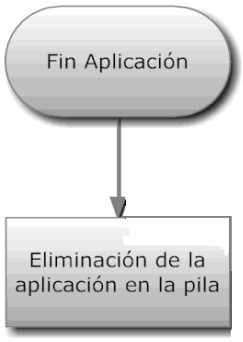


Figura 40. Diagrama de estados del caso de uso *Log out*

Funcionamiento completo de la aplicación.

Durante el funcionamiento de la aplicación, si se produjese algún problema interno tanto de la aplicación, como del dispositivo, la aplicación lanzará un mensaje por pantalla indicando que se ha producido un problema y que es necesario cerrar la aplicación.

3.2.3 Localizador.

3.2.3.1 Contexto

Se trata de la segunda parte del sistema. Es la que va destinada a los clientes que precisen conocer su posición dentro de la planta del edificio.

La aplicación se llamará localizador y estará compuesta por tres actores:

- Dispositivo móvil: su misión es la de mostrar la posición en el mapa del usuario en cada momento.

  Su tarea principal es la de realizar los barridos de señales periódicamente para obtener los niveles de potencia de las señales. Una vez realizada la acción y obtenidos los datos pertinentes se calculará la posición y se mostrará por la pantalla.

  Para que el sistema sea en tiempo real, se realizarán barridos de señal periódicamente. El periodo ha de ser lo suficientemente corto como para poder detectar el movimiento de una persona que se encuentra andando. En el capítulo de pruebas se muestra el estudio realizado, pero el tiempo de periodo estimado como válido debe estar entre los 500 milisegundos y el segundo.

- Puntos de acceso: en este caso son actores pasivos, cuya misión es ceder información al dispositivo.

  La información que la aplicación requiere de cada punto de acceso es el número MAC. El dispositivo calcula la potencia de señal que recibe y almacena ambos datos de todos los puntos de acceso que le interesan.

  Los puntos de acceso están distribuidos por la planta y en función del lugar de localización del usuario se recibirá mayor o menor intensidad de señal. La distribución de estos puntos es importante para que la cobertura de localización sea lo mayor y más precisa posible.

- Servidor web: este es el tercer actor de la aplicación y su misión es la de aportar el mapa de potencia y el mapa físico de la planta de edificio.

  Los mapas se descargarán cada vez que el usuario inicie la aplicación. Es importante que ambos ficheros no sean muy pesados para conseguir que esta descarga sea lo más ágil posible.

3.2.3.2 Casos de uso.

Los casos de uso correspondientes a la aplicación son los siguientes y se muestran en la figura 41.

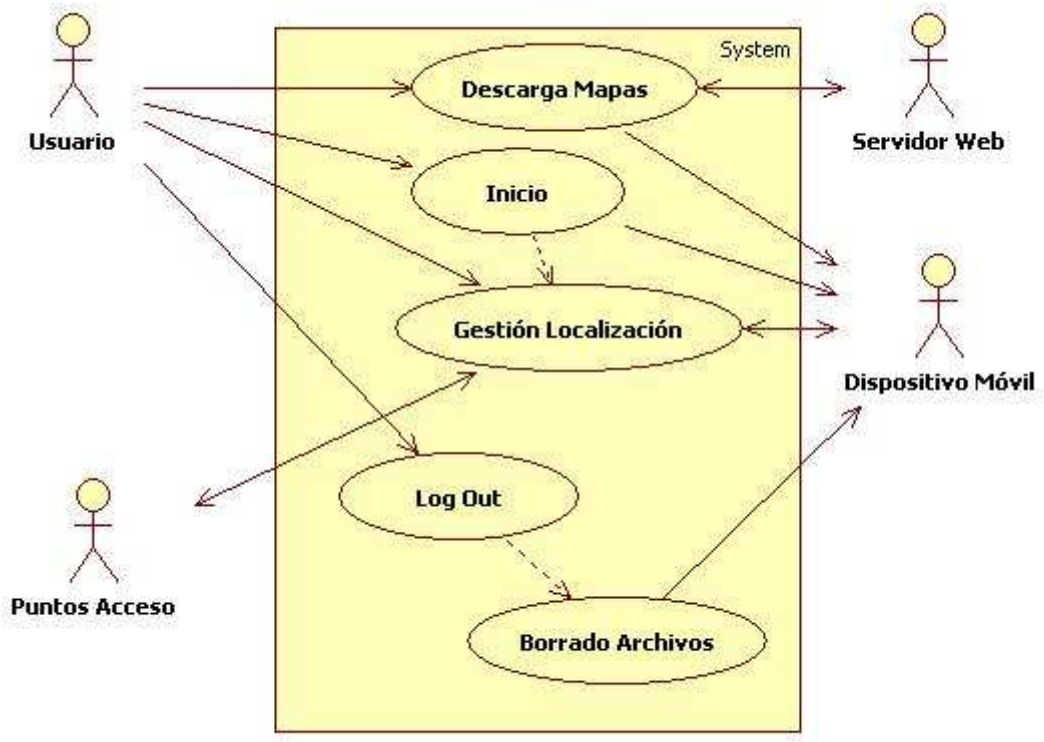


Figura 41. Diagrama de casos de uso de la aplicación Localizador.

- Caso de uso 1. Inicio.

  Con este caso de uso se arranca la aplicación de localización. Se muestra una pantalla de bienvenida en la que se muestra información sobre la aplicación que se está ejecutando, la versión de la misma y cuanta información sea necesaria (Versión de la aplicación, edificio donde presta servicio, logotipo de la empresa…).

  Esta pantalla aparecerá durante unos segundos y dará paso a la siguiente ventana, que será la descarga de los mapas.

- Caso de uso 2. Descarga de mapas.

  En este caso de uso se procederá a la descarga de los mapas. Se trata de dos archivos necesarios para el funcionamiento de la aplicación.
  La descarga se realizará de un servidor web proporcionado por el edificio donde se alojarán ambos archivos.

  Sin estos mapas el funcionamiento de la aplicación no es posible, ya que se trata del mapa de la planta del edificio y del mapa de potencias.

- Caso de uso 3. Gestión de la localización.

  Se trata del caso de uso con mayor funcionalidad de la aplicación ya que es el encargado de realizar los cálculos de la posición y de mostrarla por pantalla.

  La localización se realizará periódicamente, para poder detectar la posición del dispositivo en todo momento, siempre que éste se encuentre dentro de la zona de cobertura de la red.

- Caso de uso 4. Borrado de archivos.

  En este caso de uso lo que se trata es de mantener limpia la memoria de los usuarios, por tanto, cada vez que dejan de ejecutar la aplicación, antes del cierre de ésta, el programa eliminará los dos archivos almacenados al inicio de la ejecución.

  Así se evita duplicar la información en el caso en que produzcan varias conexiones en distintos momentos.

También se consigue que los usuarios no accedan a la ejecución del programa con material no actualizado, lo cual desembocaría en fallos en la estimación de la posición.

- Caso de uso 5. *Log out*.

  Es el caso de uso encargado de cerrar la aplicación cuando el usuario considere oportuno hacerlo.

  Una vez se inicie la operativa de cierre de la aplicación se llamará procederá al borrado de los archivos descrito en el caso de uso anterior.

3.2.3.3 Diseño lógico.

Para analizar el diseño lógico se va presentar la estructura estática y los escenarios de ejecución de la aplicación.

3.2.3.3.1 Estructura estática del sistema.

Esta aplicación tiene tres grandes bloques funcionales. A partir de la combinación de estos se logra un correcto funcionamiento para la localización.

Estos tres grandes bloques son:

- *Downloader*: se trata del bloque inicial y en el se realiza la descarga de archivos y el almacenamiento de los mismos en la tarjeta de memoria del dispositivo.

- Gestor Wi-Fi: es el bloque encargado de hacer la consulta del nivel de potencia de cada punto de acceso de la red.

- Calculador: la misión de este bloque es la de recoger e interpretar los datos de los niveles de potencia del bloque anterior para estimar la posición en el plano.

  Además, después de estimar la posición se encargará de reflejarla en el mapa por la pantalla del dispositivo.

En las siguientes tablas se exponen las distintas responsabilidades de cada bloque, así como sus colaboradores.

Bloque	*Downloader*
Responsabilidades	Presentar la pantalla de inicio de la aplicación con la información corporativa. Gestionar la descarga de los archivos necesarios para el correcto funcionamiento del programa. Estos archivos son el mapa de planta y el mapa de potencias. Una vez realizada la descarga se desconectará de la red Wi-Fi a la que estuviese conectado. Una vez finalizado su proceso dará paso a la parte de localización
Colaboradores	Gestor Wi-Fi
Notas	No puede haber dos conexiones Wi-Fi simultáneas, ya que la tarjeta de red no lo soporta.
Incidencias	

Tabla 11. Bloque *Downloader*.

Bloque	Gestor Wi-Fi
Responsabilidades	Realiza periódicamente barridos de señales Wi-Fi con el fin de obtener la potencia de la señal de los puntos de acceso Wi-Fi de la red usada para la localización. Además, de realizar el barrido guarda los datos en el formato correcto para que la aplicación pueda disponer de ellos en el momento en que proceda a hacer el cálculo de la posición.
Colaboradores	*Downloader* Calculador
Notas	Para su correcto funcionamiento no puede haber ninguna conexión Wi-Fi abierta ni estar produciéndose dos barridos de señales simultáneos.
Incidencias	

Tabla 12. Bloque Gestor Wi-Fi.

Bloque	Calculador
Responsabilidades	Se encarga de estimar la posición en la que se encuentra el dispositivo. Para ello realiza una interpretación de los datos obtenidos por el bloque Gestor Wi-Fi. Por otro lado, una vez realizada la estimación de la posición se procederá a mostrarla por pantalla, pintando un punto en el mapa.
Colaboradores	Gestor Wi-Fi
Notas	
Incidencias	

Tabla 13. Bloque Calculador

3.2.3.3.2 Escenarios de la aplicación.

Inicio.

En este escenario se muestra la pantalla de inicial de la aplicación. Esta pantalla aparecerá durante unos segundos y dará paso a la pantalla en la que se desarrollará el trabajo principal de la aplicación.

Este escenario de inicio es en sí mismo una actividad, que cuando finaliza se queda en pausa esperando que el liberador de memoria interno lo elimine o que se vuelva a activar por una orden del usuario al manejar el dispositivo.

Este escenario responde al siguiente diagrama de estados.

Figura 42. Diagrama de flujo de Inicio

Descarga.

Este escenario es donde se produce la descarga de archivos desde el servidor web. Esta descarga se produce tras la pantalla inicial. Durante el tiempo que dura la descarga aparecerá una barra de progreso, indicando que se está procediendo a la descarga de archivos necesarios para el funcionamiento de la aplicación.

Se realizará la descarga de dos archivos que se alojarán en el directorio raíz de la tarjeta de memoria externa del dispositivo. Estos archivos serán una imagen en un formato soportable por el dispositivo (jpg, gif, png…) y un archivo de texto.

Tras realizar la descarga se pasará a la siguiente pantalla o actividad, que será la de estimar la localización del individuo en el edificio.

A continuación se muestra un diagrama de flujo del escenario:

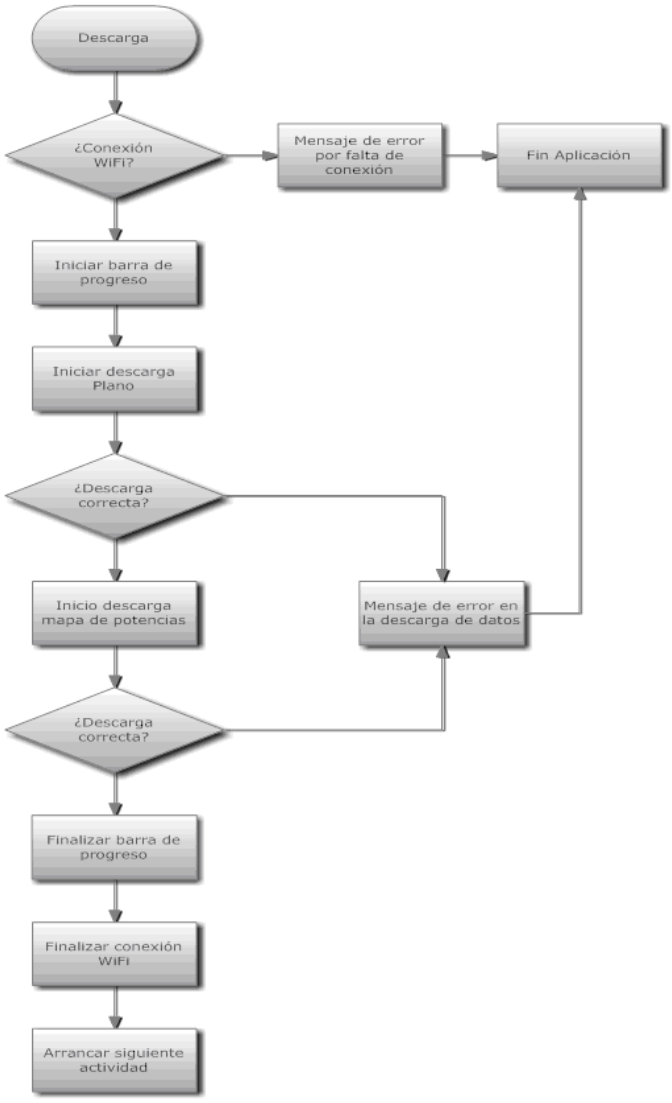


Figura 43. Diagrama de flujo de Descarga.

Localización.

Se trata del escenario principal de la localización. Se compone de tres partes: obtención de las intensidades de los puntos de acceso de la red, cálculo de la posición y representación de la estimación en el mapa.

En la primera parte el programa se encuentra a la espera del primer sondeo de las redes, a partir de ese momento cada barrido se realiza periódicamente. Una vez hecho el barrido se obtiene la información que interesa para la localización, es decir, los números MAC de cada punto de acceso de la red detectado y su nivel de potencia. Una vez transformados los valores obtenidos y almacenados se pasa a la siguiente fase, la estimación de la posición.

En esta segunda fase, como se ha dicho en el párrafo anterior, se procede al cálculo de la posición. Para ello se van buscando los valores obtenidos en el escáner en el mapa de potencias. No se busca el resultado exacto ya que las señales pueden estar influenciadas por factores externos, así que se comparan los valores empíricos con los valores teóricos, estos últimos dentro de un margen de sensibilidad. Este margen de sensibilidad es de ± 5 dBm, y se calcula en el capítulo de pruebas.

Finalmente,, cuando se detecta la posición la aplicación pasa la fase de representación del resultado obtenido. En este momento obtenemos los píxeles de la posición en la que nos encontramos del mapa de potencias, y almacenando ambas coordenadas en el programa. A continuación se invoca el método para pintar el punto en el mapa y se nos muestra por pantalla.

Estas tres fases se realizan periódicamente hasta que el usuario decide finalizar la aplicación o el dispositivo se encuentra tan comprometido de memoria como para cerrar una aplicación.

En la siguiente figura podemos ver el diagrama de flujo de este escenario:

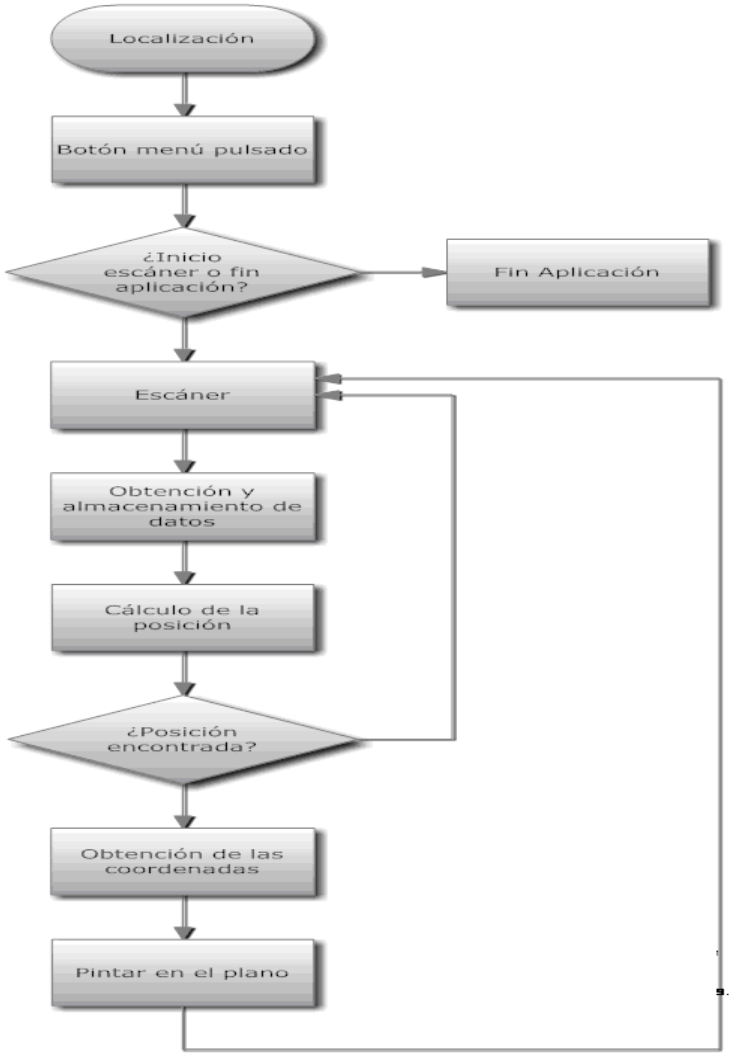


Figura 44. Diagrama de flujo de Localización.

Reinicio de la localización.

Existe la posibilidad de reiniciar la localización de la aplicación sin tener que salir de la aplicación.

Para reiniciar la localización se pulsa el botón del menú de la aplicación y de los botones que aparecen el usuario debe elegir la opción de reiniciar el escáner.

El siguiente diagrama de flujo, muestra esta operativa:

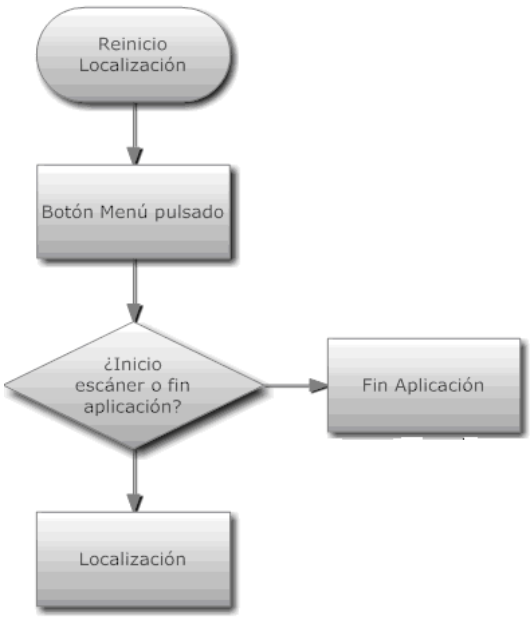


Figura 45. Diagrama de flujo de Reinicio Localización.

Cierre de la aplicación.

En esta parte de la aplicación se finaliza el programa. Para ello se debe pulsar el botón de menú de la aplicación y seleccionar la opción de finalizar programa.

Al finalizar la aplicación se procederá al borrado de los archivos descargados en el escenario inicial, evitando así la acumulación de versiones antiguas y dejando la memoria del dispositivo del usuario de la misma manera que la tenía al inicio de la ejecución del programa.

En siguiente diagrama se puede apreciar el flujo que sigue la aplicación en este escenario.

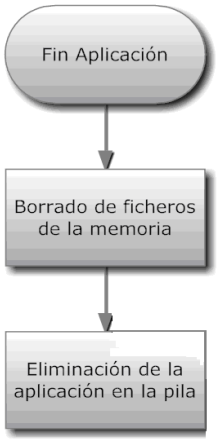


Figura 46. Diagrama de flujo del fin de la aplicación.

Funcionamiento completo de la aplicación.

Durante el funcionamiento de la aplicación, si se produjese algún problema interno tanto de la aplicación, como del dispositivo, la aplicación lanzará un mensaje por pantalla indicando que se ha producido un problema y que es necesario cerrar la aplicación.

3.3 Implementación.

3.3.1 Lenguaje de programación y entorno de desarrollo.

El lenguaje de programación sobre el que se desarrolla en Android es Java. Sin embargo, también soporta aplicaciones escritas en otros lenguajes como C/C++ y estas pueden ser compiladas a código nativo de ARM y ser ejecutadas normalmente. Aunque los programas de Android se escriben en Java, los archivos y librerias de nivel más bajo están escritas en lenguaje C/C++.

Para programar en Android es necesario descargar de Internet el SDK propio de Android. Este SDK es el kit de desarrollo necesario para programar e implementar todo tipo de aplicaciones para Android y el sistema operativo para teléfonos móviles propuesto por Google.

Este paquete de desarrollo incluye las APIs y herramientas necesarias para desarrollar las aplicaciones utilizando JAVA como lenguaje de programación.

3.3.2 El entorno de desarrollo.

Para programar en la plataforma Android se necesita que estén presentes cuatro archivos que están relacionados entre sí ya que, en mayor o menor medida, unos dependen de los otros. Estos archivos son: una archivo xml para la interfaz gráfica, el *AndroidManifest*, que también es un archivo xml, el archivo .class donde se desarrolla la aplicación y un archivo R.Java que lo genera la aplicación automáticamente y sirve como abreviación a la hora de referenciar cualquier recurso en los xml anteriores. Para escribir los archivos se usa el IDE de Eclipse.

Para facilitar la programación, Java permite organizar las clases de una forma más o menos ordenada usando paquetes ("Bibliotecas"). De esta manera, se pueden reciclar los paquetes de una aplicación a otra para que puedan ser reutilizados. De hecho el JDK incluye paquetes básicos de Java que facilitan la programación como son por ejemplo, la salida por pantalla, el manejo de fechas, etc.

Además, de las bibliotecas ofrecidas por Java, para programar en Android se necesitan paquetes desarrollados por Google. Éstos están en el SDK de Android y permiten acceder a funciones de la plataforma como los mensajes, la libreta de direcciones, el GPS, etcétera.

Las aplicaciones de Android se ejecutan en una máquina *virtual* propia desarrollada por Google y se compilan para esta máquina. Para ello el SDK de Android trae un "traductor" de archivos .class a archivos .dex (los que necesita).

Gracias a que Eclipse permite extender su funcionalidad a través del *plug-in* de Android, creando una configuración de proyecto de tal forma que tan sólo hay que preocuparse por escribir, ya que el se encarga de preparar el entorno con una configuración de proyecto que permite compilar/ejecutar de forma similar a la que se hace con proyectos Java normales.

3.3.3 Decisiones de Implementación.

3.3.3.1 Toda la operativa en el dispositivo

Existían dos posibilidades a la hora de realizar los cálculos de la posición del individuo. La primera opción consistía en enviar los datos obtenidos de las potencias de las señales a un servidor y que éste hiciese todos los cálculos necesarios para determinar la posición. Una vez finalizados los cálculos, se enviaba el resultado al dispositivo.

La segunda opción es recoger y analizar los datos en el terminal, sin introducir ningún elemento intermediario.

Se ha optado por esta segunda opción por varias razones. La primera y principal razón es que los terminales en los que se ha montado el sistema operativo Android, al menos hasta el momento, no se pueden realizar dos operaciones que requieran la tarjeta de red a la vez. Es decir, no se puede es*table*cer una conexión Wi-Fi con un servidor y a la vez estar realizando escáneres de las señales.

Otra de las razones por las que se opta por desarrollar el posicionamiento en el terminal es que el consumo de batería es muy elevado en transmisión y recepción de datos. Esto puede provocar que se agote la batería del dispositivo demasiado rápido. Además, lo que se podría conseguir enviando los datos a un servidor es que la velocidad de los cálculos fuese mayor, pero los cálculos que requiere este sistema son sencillos y el dispositivo está perfectamente capacitado para realizarlos en un tiempo aceptable.

A continuación, en el capítulo de pruebas y resultados se muestran los resultados de los estudios realizados para el consumo de batería del programa en el terminal HTC Dream.

3.3.3.2 Método de detección de la posición.

El método elegido es del vector de potencias ya que este sistema es el que mejor se adapta tanto a las condiciones de la red como a la información que puede obtener del dispositivo.

Se descarta la triangulación de potencia porque para obtener la posición tendríamos que comprobar en qué puntos se cortan los círculos de tres señales y a partir de ahí resolver un sistema de ecuaciones.

En la figura 47 se ven los tres posibles casos que se pueden dar en la triangulación de señales.

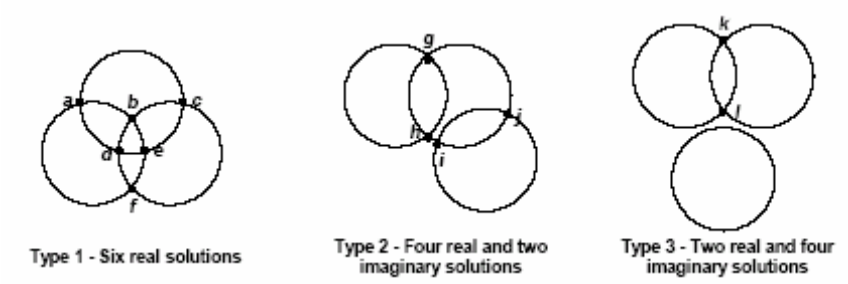


Figura 47. Posibles intersecciones en la triangulación de una señal.

Este sistema no es válido debido que en un entorno como el que forman los pasillos de una planta de un edificio no se obtienen círculos de potencia como los de la figura debido a los numerosos rebotes, reflexiones y difracciones que sufre la señal en su trayecto. Este hecho podría dar lugar mediciones erróneas de la posición.

Además, el coste computacional de los cálculos supone un gran consumo de energía del terminal, ya que habría que resolver el sistema de ecuaciones mencionado anteriormente de manera iterativa, lo que provocaría un aumento considerable del número de dichos cálculos.

También se ha descartado la heurística de proximidad para determinar la posición del usuario ya que esta técnica se basa en encontrar el punto de acceso más cercano a la posición del dispositivo y a partir de ahí comenzar los cálculos. Esta técnica no es viable debido a la disposición de los puntos de acceso, ya que éstos no se encuentran distribuidos uniformemente por la planta si no que se encuentran en un pasillo.

La heurística de los vecinos más cercanos también se descarta por el nivel de complejidad que alcanzaría el sistema, ya que son necesarios un gran número de puntos de calibración para hacer las comparaciones y el aprendizaje.

Por último, si se tiene en cuenta la heurística de movimiento, realizando un entrenamiento en los puntos de control tanto en movimiento como en estático. Aunque a priori el movimiento es un factor importante, a la velocidad de una persona andando no lo es tanto, como se demostrará en el capítulo de pruebas.

3.3.3.3 Interfaz gráfica.

Aunque técnicamente es posible crear y adjuntar los *widgets* para nuestra actividad tan solo a través de Código Java, el enfoque más común es utilizar *scripts* basados en un diseño XML. La llamada dinámica de instancias de *widgets* se reserva para situaciones más complejas, donde los *widgets* no se conocen en tiempo de compilación.

Como su nombre indica, un diseño basado en XML es una especificación de *widgets*, las relaciones de cada uno de ellos con el resto y con sus contenedores, todo esto codificado en formato XML.

¿Por qué usar diseños basados en XML?

La mayoría de todo lo que se puede hacer en archivos de diseño de XML se puede lograr a través de Java. Por ejemplo, se podría utilizar `setTypeface()` para tener un botón que tuviese su texto en negrita, en lugar de utilizar una propiedad en un esquema XML.

Quizás la razón más importante para usar una interfaz gráfica basada en XML es la ayuda que se obtiene gracias a la creación de herramientas para la definición de la vista de la aplicación. Una de estas herramientas es el constructor de GUI para el entorno de desarrollo de Eclipse. Otra herramienta que se puede encontrar en internet es DroidDraw[1].

Si la información se encuentra estructurada y ordenada es más sencillo crear la interfaz gráfica en un fichero XML, que en código Java.

Por otro lado, al estar separado la parte gráfica del código de Java facilita la labor a posibles desarrolladores secundarios, que trate de reinterpretar y modificar el código.

Además, XML como un formato de definición de interfaz gráfica de usuario es cada vez más común. Como ejemplos tenemos XAML2 de Microsoft, Flex3 de Adobe, y XUL4 de Mozilla. Todos tratan de adoptar un enfoque similar al de Android: poner el diseño gráfico en un archivo XML.

[1] http://www.droiddraw.org/

3.3.3.4 Formato de los archivos

En el caso del mapa de potencias se ha optado por elegir un archivo de texto simple, sin formato. La elección de este archivo se debe a que de los editores de texto sus archivos son los de menor tamaño, con diferencia. Se ha elegido un formato de fichero de texto, ya que el usuario podría tener la necesidad de consultar los resultados obtenidos por los escáneres. Un ejemplo de este archivo sería las líneas que se muestran a continuación, habiendo entre cada zona un salto de línea:

```
Zona1 00:0d:54:99:7a:42 -77 00:c0:ca:19:f1:09 -75 06:c0:ca:19:f1:09 -75
06:c0:ca:19:f0:fc -91 06:c0:ca:19:f1:1b -77 00:c0:ca:19:f1:40 -76
06:c0:ca:19:f1:40 -77 00:c0:ca:19:f0:fc -91 06:c0:ca:19:f1:06 -95
00:c0:ca:19:f1:1b -77 89 104
Zona2 00:0d:54:99:7a:42 -88 06:c0:ca:19:f1:40 -66 00:c0:ca:19:f1:09 -90
06:c0:ca:19:f1:09 -90 00:c0:ca:19:f1:1b -90 06:c0:ca:19:f1:1b -90
00:c0:ca:19:f0:fc -90 06:c0:ca:19:f0:fc -91 06:c0:ca:19:f1:06 -90
00:c0:ca:19:f1:06 -91 00:c0:ca:19:f1:40 -66 83 153
Zona3 06:c0:ca:19:f1:40 -59 00:c0:ca:19:f1:40 -58 06:c0:ca:19:f1:06 -88
00:c0:ca:19:f1:06 -88 00:c0:ca:19:f0:fc -88 06:c0:ca:19:f0:fc -88
06:c0:ca:19:f1:1b -95 00:c0:ca:19:f1:1b -95 80 180
Zona4 00:c0:ca:19:f1:40 -64 06:c0:ca:19:f1:40 -64 00:c0:ca:19:f1:06 -92
06:c0:ca:19:f0:fc -77 00:c0:ca:19:f0:fc -77 06:c0:ca:19:f1:06 -94
00:c0:ca:19:f1:1b -94 83 207
Zona5 06:c0:ca:19:f1:40 -63 00:c0:ca:19:f1:40 -63 06:c0:ca:19:f0:fc -67
00:c0:ca:19:f0:fc -67 06:c0:ca:19:f1:5f -85 00:c0:ca:19:f1:5f -85
06:c0:ca:19:f1:06 -93 00:c0:ca:19:f1:06 -94 06:c0:ca:19:f1:1b -96
00:c0:ca:19:f1:1b -94 91 233
```

El formato de línea es el siguiente:

<Zona X> <Mac 1> <Potencia1> <Mac2> <Potencia2>….<Píxel X> <Píxel Y>

En la figura 48 se muestran los pesos para el mismo mapa de potencia usado en la aplicación en distintos formatos de texto:

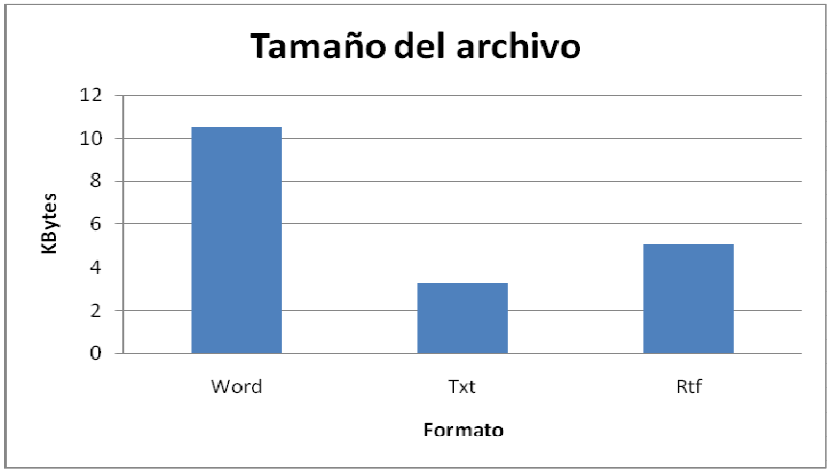


Figura 48. Pesos del mapa de potencias en función del tipo de archivo.

En cuanto al plano de la planta del edificio se ha optado por un archivo jpg aunque bien podría haber sido un archivo png, ya que ambos son soportados por el dispositivo y no existen grandes diferencias entre uno y otro.

A la hora de guardar los archivos, Android ofrece la posibilidad de guardarlos en la propia aplicación, lo cual resulta muy útil para tener todo ordenado y almacenado en el mismo paquete del programa. Sin embargo, el problema aparece cuando tras la descarga se trata de agregar estos archivos. Esto no es posible hacerlo porque el programa ya está compilado y ejecutándose, por lo que resulta imposible modificarlo en tiempo de ejecución. Ante este inconveniente se optó por añadirlos en la tarjeta de memoria externa, en el directorio raíz. Con esto se consigue disponer de los archivos siempre que se necesiten y además, al estar en el directorio raíz, no será necesario crear nuevas carpetas, lo cual puede ralentizar la aplicación.

Se ha optado por un fichero de texto y no una base de datos por las siguientes razones. La primera y principal, es que un usuario que quisiese acceder a los datos tendría que tener conocimientos de programación para poder volcar el contenido de la base de datos en un fichero. La segunda razón es que la base de datos de Android, SQLite, está implementada en bibliotecas de C y habría que crear unas nuevas bibliotecas en Java. Esto supondría un aumento del tiempo de trabajo y la complejidad del proyecto sin ser un objetivo del proyecto la creación de bibliotecas para el manejo de la base de datos.

3.3.3.5 Margen de sensibilidad en la estimación de la posición.

Se ha estimado según los estudios presentados en el capítulo de pruebas que el número de óptimo de repeticiones para el escáner de señales ha de ser de 250, ya que aporta la información suficiente y no tarda demasiado tiempo en completarse.

Además, se quedó demostrado empíricamente sobre un entorno real, que a pesar de las 250 repeticiones en el Calibrador, a la hora de calcular la posición era necesario un margen de sensibilidad de 10 dBm, es decir, 5 dBm por encima y 5 dBm por debajo del valor obtenido en el escáner de la aplicación del Localizador.

Así se logra asegurar la medida y realizar una estimación más certera de la posición en la que se encuentra el individuo.

Este margen de potencia en dBm equivale a 10 mW.

3.3.4 Diagramas de clases UML.

La figura 49 muestra el diagrama de clases de la aplicación calibrador. En esa figura quedan reflejadas las relaciones que hay entre las distintas clases de la aplicación.

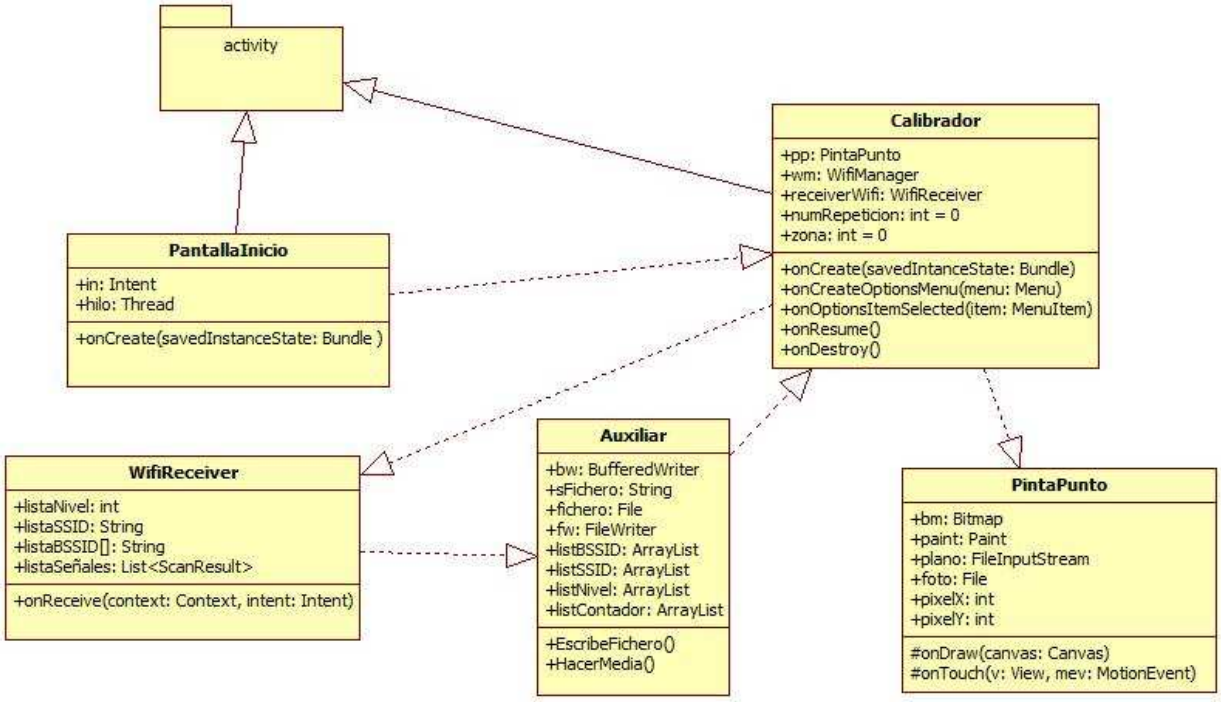


Figura 49. Diagrama UML de la aplicación Calibrador

Como se puede ver en el diagrama existen dos actividades en esta aplicación, o lo que es lo mismo dos pantallas distintas. La primera pantalla está desarrollada a través de la clase PantallaIncio que extiende de la interfaz *Activity*. En esta actividad se muestra una pantalla resumen con los datos relevantes de la aplicación y pasados unos segundos inicia la segunda actividad, Calibrador.

En Calibrador es donde se realizan todas las operaciones para conseguir el mapa de potencias. Primero se inicia el escáner de señales y éstas se almacenan por medio de la clase `WifiReceiver`. Esta clase invoca a los métodos de la clase auxiliar `HacerMedia()` y `EscribirFichero()` que son los encargados de calcular las potencias medias obtenidas y guardarlas en el mapa de potencias que se está generando. En todo momento siempre que se produzca un toque en la pantalla, mediante la clase `PintaPunto`, se procederá a la representación del punto en la pantalla.

En la figura 50 observamos a continuación el diagrama UML de clases de la aplicación Localizador.

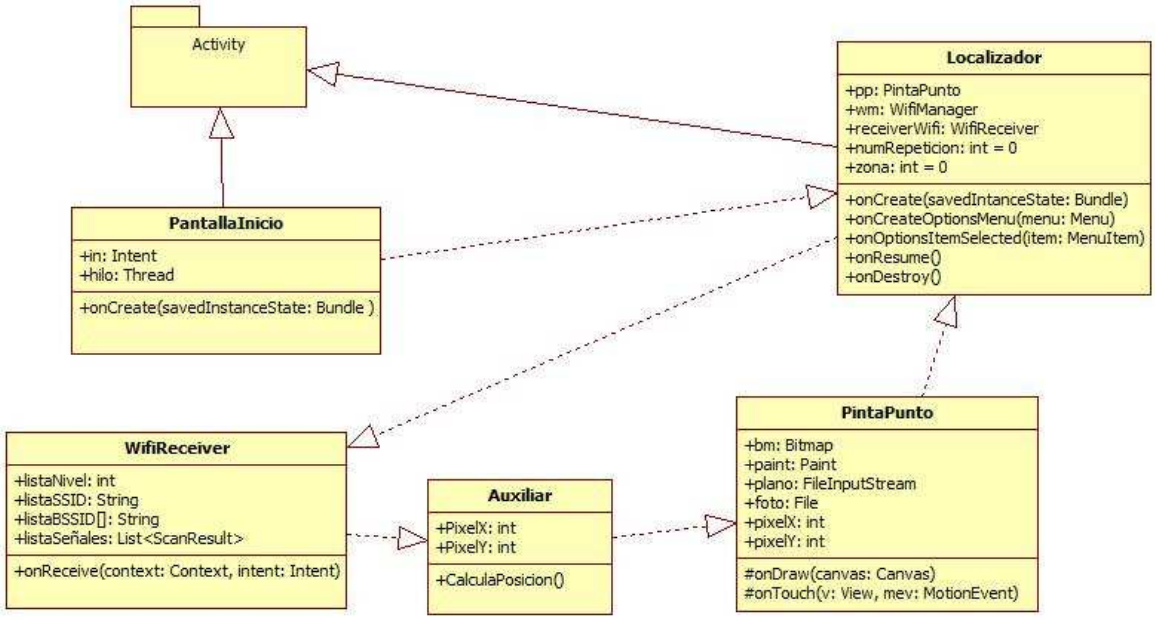


Figura 50. Diagrama UML de la aplicación Localizador

Se puede observar que al igual que en la aplicación Calibrador, en esta aplicación también se disponen de dos actividades. La primera es similar a la de la aplicación anterior, una presentación de los datos de interés de esta.

La actividad de la clase Localizador es la encargada de gestionar la representación de la posición del individuo en el plano. Para ello realiza escáneres usando la clase `WifiReceiver`, esta clase utiliza la clase Auxiliar de la aplicación para calcular la posición y pasar las coordenadas expresadas en píxeles a la clase `PintaPunto` para su representación en el plano.

Capítulo 4. Pruebas y resultados

En este capítulo se van a presentar las pruebas realizadas a las dos aplicaciones del sistema. También se presentará un análisis de los resultados obtenidos y valoración de los mismos.

4.1 Entorno de pruebas.

Las pruebas se han realizado en la primera planta del Edificio Torres Quevedo, en el campus de Leganés de la Universidad Carlos III. Éstas se han centrado en la zona del departamento de Telemática.

Para la correcta realización de las pruebas es necesario disponer de una red Wi-Fi con cobertura en toda la zona del estudio, se utilizará la red WLIT del departamento de telemática. Serán necesarios una conexión con un servidor web y un dispositivo móvil con tarjeta de red inalámbrica y sistema operativo Android instalado. En el primer caso se realizará una conexión con la red Wi-Fi-UC3M, que es gratuita y de libre acceso. En cuanto al dispositivo móvil se ha utilizado el terminal HTC Dream o G1, que cumple con los requisitos necesarios para completar las pruebas satisfactoriamente.

- Red Wi-Fi: se ha utilizado la red Wi-Fi del laboratorio de Telemática. Esta red está formada por seis *routers* cada uno de los cuales tiene dos direcciones MAC. A la hora de realizar las pruebas se comprobó, como cabía esperar, que las potencias de cada *router* eran similares y por tanto solo había que tener en cuenta una de las dos direcciones.

- Servidor web: se ha decidido utilizar el servidor web de la universidad ya que no es necesario introducir ningún nombre ni contraseña para conectarnos a la red, ya que es una red de acceso libre dentro del campus universitario. Una vez

realizada la descarga de los archivos que se necesitan para el correcto funcionamiento de la aplicación, ésta provoca una desconexión de la red.

- HTC Dream o G1. La elección de este terminal viene determinada por el hecho de que en el momento en que se inicia el desarrollo del proyecto, Android ofertaba a sus desarrolladores este dispositivo. Además, es el indicado pues cumple todas las condiciones para que en el se puedan ejecutar las dos aplicaciones desarrolladas en el proyecto.

4.2 Aplicación Calibrador.

Para probar esta aplicación se tomaron medidas en varios puntos de la zona del departamento y se comprobó que las anotaciones que se hacían en el fichero de texto eran consecuentes con lo que se pretendía en el diseño. Los puntos de medida son los que aparecen en la figura 51.

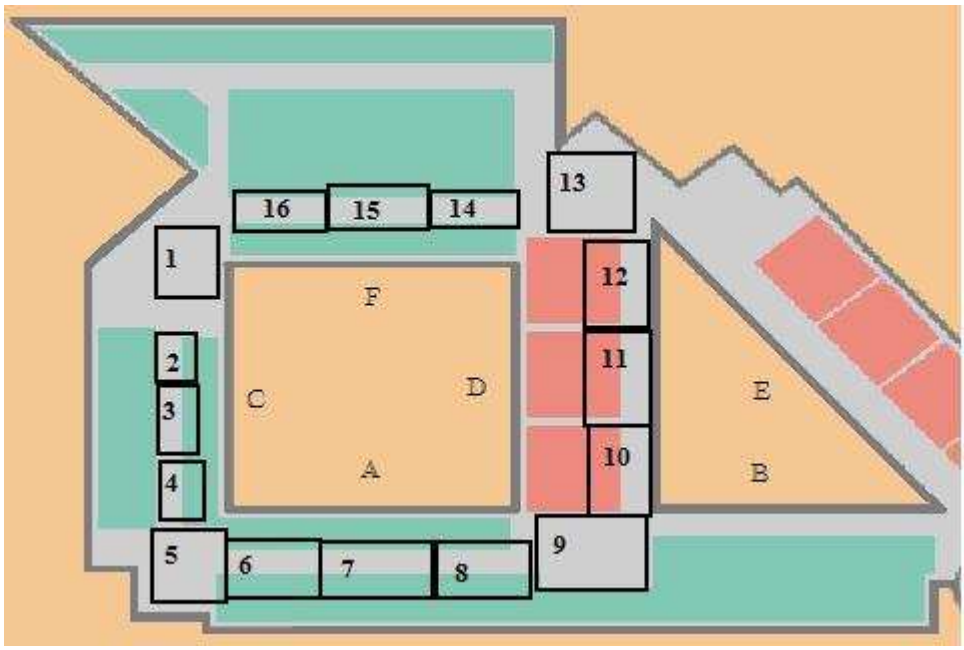


Figura 51. Sectores de la planta.

Uno de los principales objetivos de estas pruebas era comprobar el número óptimo de repeticiones del escáner de señales para cada punto. Se debía llegar a un punto de compromiso en el cual las repeticiones no fuesen tan escasas como para considerar la medida poco fiel a la realidad y tampoco fuese un número de repeticiones tan elevado como para consumir en exceso la batería del terminal.

Además, se descubrió que el tiempo que tardaba en completar las repeticiones tiene una relación lineal con el aumento de éstas. En la figura 52 se puede comprobar como el tiempo aumenta en una relación constante con el número de repeticiones.

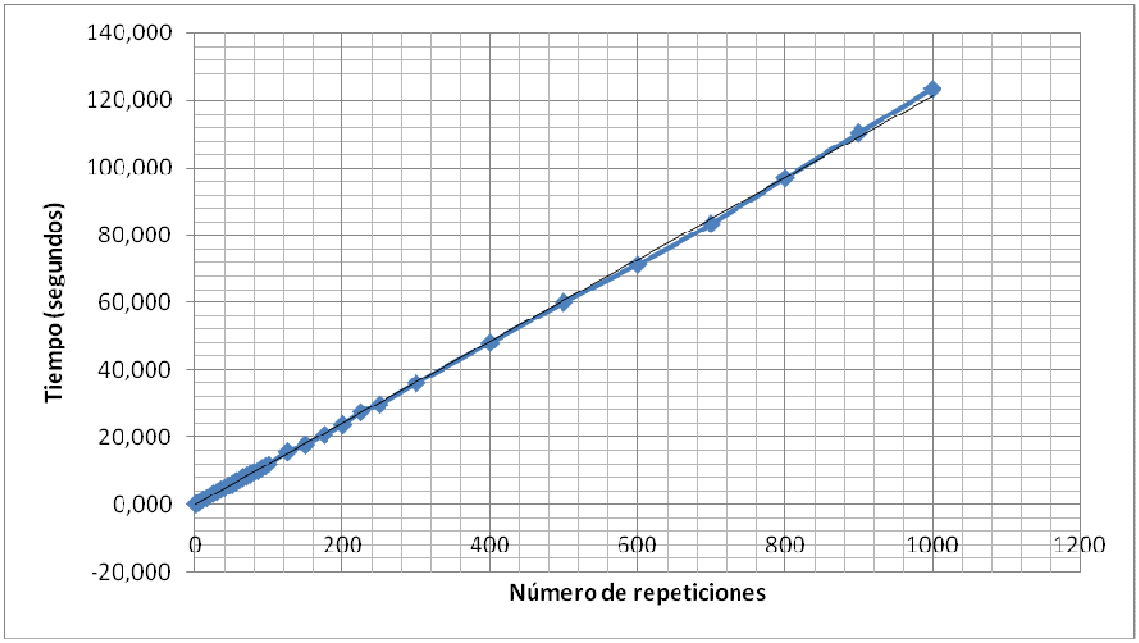


Figura 52. Gráfica que representa segundos que se tardan en realizar repeticiones del escáner de señales.

Por ello y en vista a la línea que seguía se optó por 250 repeticiones ya que no suponía un tiempo excesivo y cubría todas las posibles variaciones que se pudiesen producir, como aumentos o descensos repentinos de carga en la red, interferencias momentáneas, errores en la medida por parte del terminal, etcétera. Además, este número de repeticiones es el que mejor se ajustaba a los requisitos de consumo de batería y fiabilidad.

Para comprobar que las muestras no varían mucho en las 250 repeticiones se muestran las siguientes gráficas en la figura 53, que muestra los valores obtenidos en las repeticiones. No se han incluido los valores nulos, ya que estos no entran a formar parte del cálculo de la media.

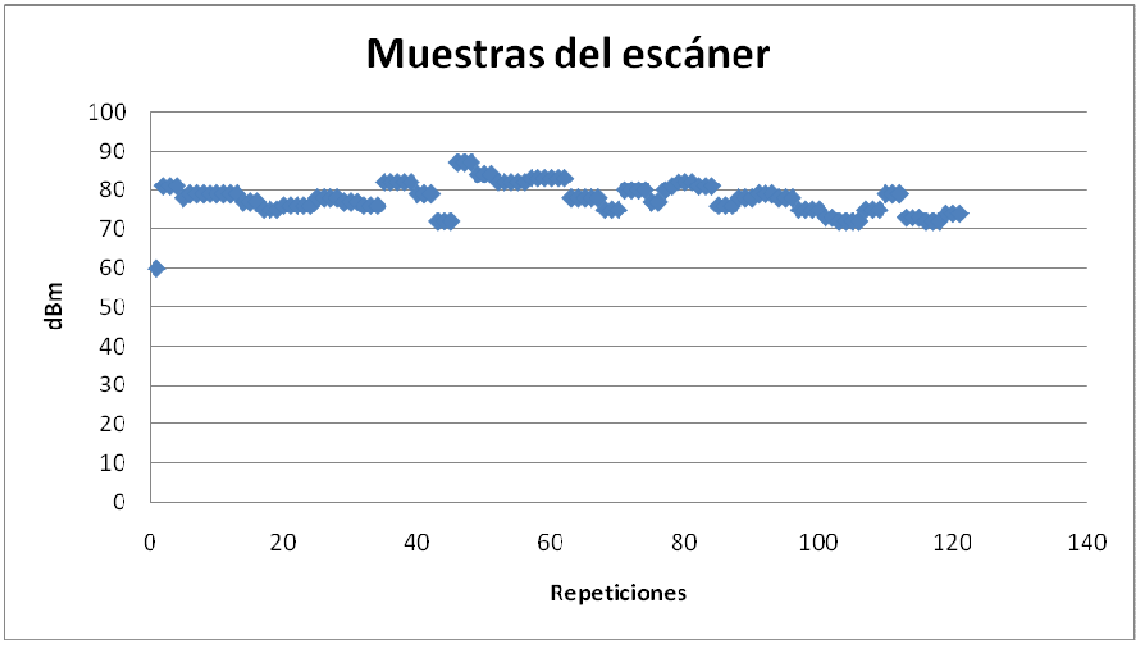


Figura 53. Potencias del escáner

En la siguiente tabla (Tabla 14) se puede ver los valores estadísticos obtenidos para este conjunto de muestras y se puede concluir que el método para obtener la potencia de referencia para cada punto de control es válido.

Moda	79
Desviación estándar	4,90974272
Min	87
Max	60
Mediana	78
Media	77,8257498

Tabla 14. Datos estadísticos de las muestras tomadas en un escáner

En la figura 54 se puede observar el tiempo que se tarda en realizar 10 repeticiones de un escáner. En el recuadro superior se muestra el tiempo en el que empieza el primer escáner y en el recuadro inferior se ve el tiempo en el que finaliza el último.

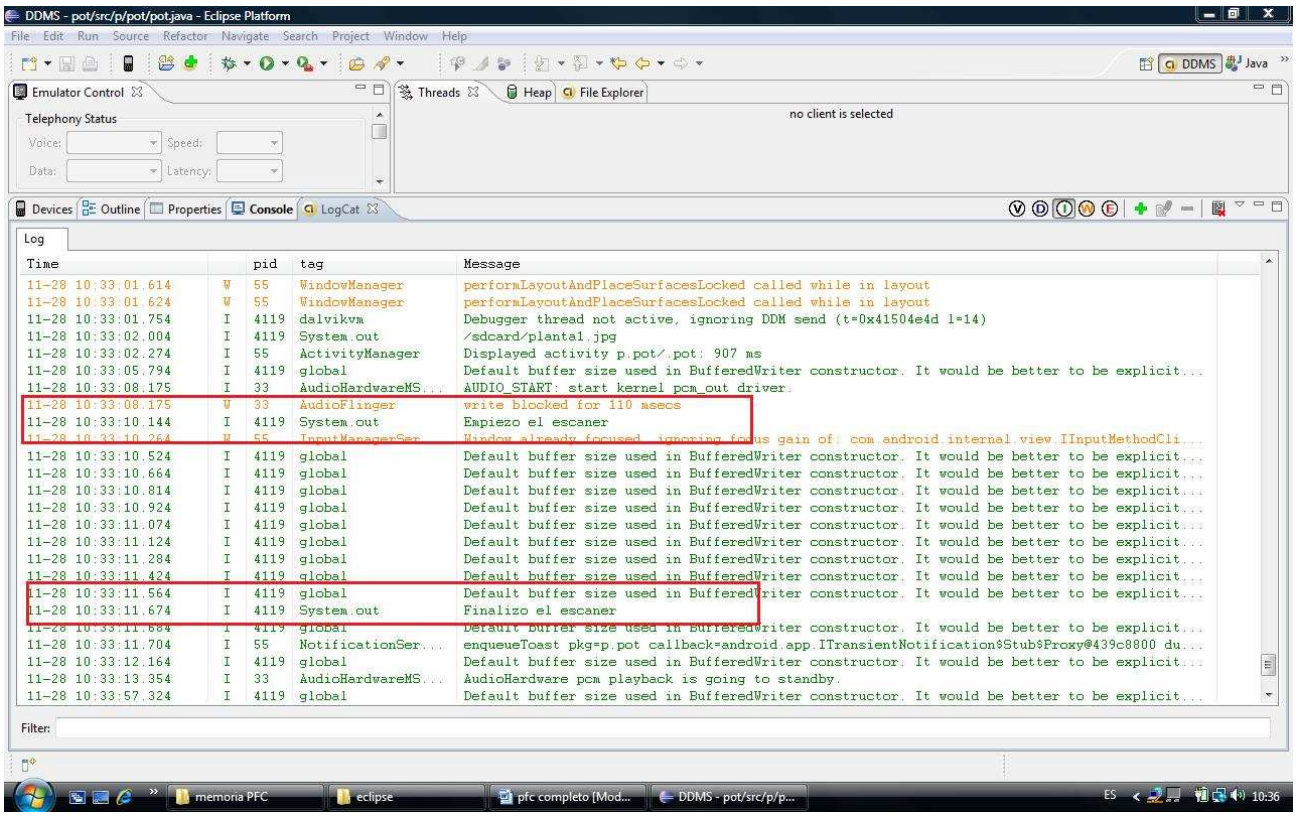


Figura 54. Tiempo que se tarda en realizar 10 escáneres.

Otro de los objetivos de las pruebas era conseguir de manera sencilla reflejar la posición. El problema sobrevino cuando para indicar la posición en el mapa no se pudo utilizar un lapicero común para pantallas táctiles, ya que en las pantallas capacitivas el elemento que se usa es el dedo. Esto suponía un inconveniente porque al pulsar con el dedo sobre el mapa no siempre se acertaba en la posición que se pretendía. Esto es debido a que en una pulsación con el dedo el terminal interpreta que se han pulsado una malla de píxeles y con el píxel que se quedaba la aplicación era el último en contactar con el dedo. En la figura 55 que se muestra a continuación podemos ver que se seleccionaba una malla de píxeles.

En el recuadro rojo se encuentran señalados los píxeles pulsados a tocar la pantalla con el dedo índice.

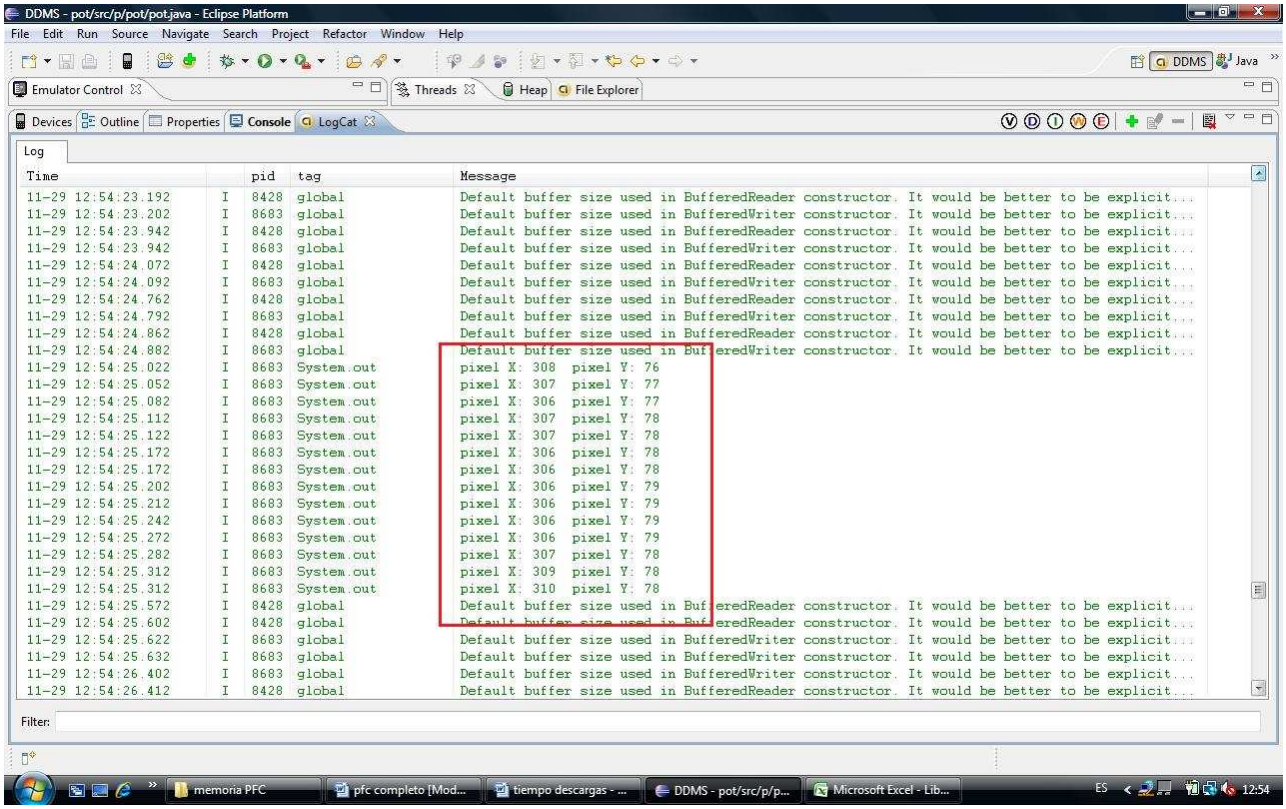


Figura 55. Malla de píxeles que se pulsan con un dedo.

Esto supuso un problema pues la idea inicial era que en cada pulsación se guardasen las coordenadas y se hiciese el escáner automáticamente. Para solventar el inconveniente se decidió que había que dar al usuario tantas oportunidades como este considerase necesarias para colocar el punto de control en el mapa. Por ello se añadió el menú para que el escáner fuese controlado también por el usuario, el cual se muestra en la figura 56.

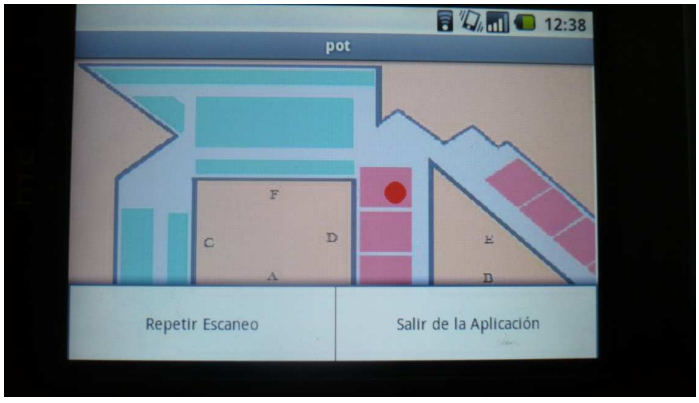


Figura 56. Vista del menú de la aplicación Calibrador.

Otro de los datos que se tuvo que estudiar empíricamente fue si realmente era necesario tener en cuenta la heurística de movimiento en el algoritmo del vector de potencias para calcular la posición del individuo. Como se explicó en capítulo del estado del arte, en la heurística del movimiento consistía en tener en cuenta, a la hora de realizar las medidas de las potencias, si el usuario se encontraba en movimiento o parado. En base a los resultados obtenidos empíricamente en un entorno real, se desestimó la idea de atender a esta heurística, ya que el tiempo que tarda en obtener los datos de un escáner es muy pequeño en comparación con la velocidad de un ser humano andando.

El tiempo que se tarda en realizar un barrido de la señal son 0,11 segundos, en este tiempo una persona caminando a paso ligero (4 Km/h) habría recorrido 12 centímetros aproximadamente, lo que no supone un gran cambio en la posición del usuario.

Estas dos eran las principales incidencias funcionales de la aplicación, que determinaron el diseño de la aplicación.

4.3 Aplicación Localizador.

Las pruebas realizadas en esta aplicación pretendían estimar los mejores valores de sensibilidad en la captación del nivel de señal de cada punto de acceso, la periodicidad de los escáneres y dónde llevar a cabo los cálculos de la posición.

La primera de las decisiones importantes en esta aplicación era donde realizar el cálculo de la posición. Por un lado se tenía la posibilidad de realizar los cálculos en un servidor remoto y por otro lado los cálculos se podían hacer directamente en el terminal. Utilizando la primera opción se tendría que producir una comunicación continua entre el servidor y el dispositivo, con el consecuente aumento en el consumo de batería. Además, se tendría que conectar y desconectar cada vez que se enviasen los datos pues la tarjeta de memoria no está preparada para realizar dos operaciones simultáneamente, lo que supondría un aumento intolerable en el cálculo de la posición.

Para demostrar esto se realizó un estudio del tiempo que se tardaba en realizar una conexión a la red inalámbrica, un envío de un fichero de texto con los datos del escáner, una desconexión y otra conexión y finalmente un envío de otro fichero de datos con las coordenadas y se obtuvo que se tardaría en realizar todo el cálculo de la posición, sin tener en cuenta el tiempo que se invertiría en resolver el algoritmo de cálculo, un tiempo de aproximadamente 11 segundos.

Esta prueba se realizó en una red sin ningún dispositivo más conectado, lo que quiere decir que si aumentamos el número de dispositivos en la red realizando descargas y subidas, este tiempo aumentaría.

Además, el hecho de realizar descargas y subidas de la red también afectaría al consumo de batería. A este consumo habría que sumarle el gasto de batería originado por el envío y la descarga de datos del servidor donde se calcularía la posición. Según el estudio realizado utilizando el programa que aparece en el anexo C, tras 10.000 peticiones de escaneo de la red, sin que se produjese ninguna descarga de archivos, la batería disminuyó un 84%. Los 10000 escáneres equivaldrían a estar utilizando el programa durante casi tres horas.

Para calibrar cuál era la sensibilidad mínima para lograr una estimación satisfactoria se realizaron pruebas empíricas con distintos márgenes. Las pruebas consistían en ver los resultados de la estimación sobre el plano en el edificio. Se probaron márgenes desde 0 dBm hasta ± 10 dBm y se comprobó que el margen en el que se obtenían mejores resultados era el ±5 dBm, aun no siendo perfectos. Con 0 dBm se tardaba mucho en encontrar la posición y con ±10 dBm confundía la posición con otras posiciones.

El siguiente dato que se tenía que decidir era el de la periodicidad de los escáneres. Se comprobó que utilizar una periodicidad inferior a 500 milisegundos, provocaba errores en la estimación de la posición pues los puntos de control estaban separados, a la velocidad de una persona andando, un tiempo aproximado de 1,5 segundos. Sin embargo, si se escogían valores superiores al segundo también se realizaba una estimación incorrecta, pues sumado al tiempo que se tardaba en realizar los cálculos de la posición, entre 400 y 600 milisegundos, superaba la distancia temporal entre los puntos de control. Como se ha comentado anteriormente la velocidad de una persona andando es de uno 4Km/h.

4.4 Vistas de las aplicaciones.

Finalmente, el resultado obtenido del desarrollo de las aplicaciones es el que se muestra en las siguientes fotografías del dispositivo:

- Calibrador.

  En la figura 57 se muestra la pantalla de inicio de la aplicación.

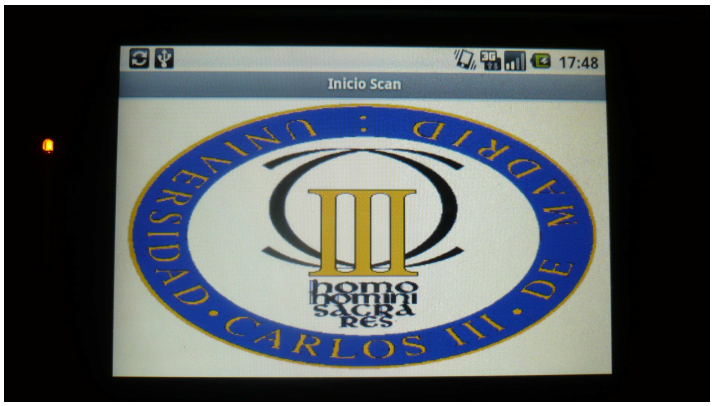


Figura 57. Pantalla Inicial Calibrador

  En la figura 58 se observa el mapa con un punto seleccionado.

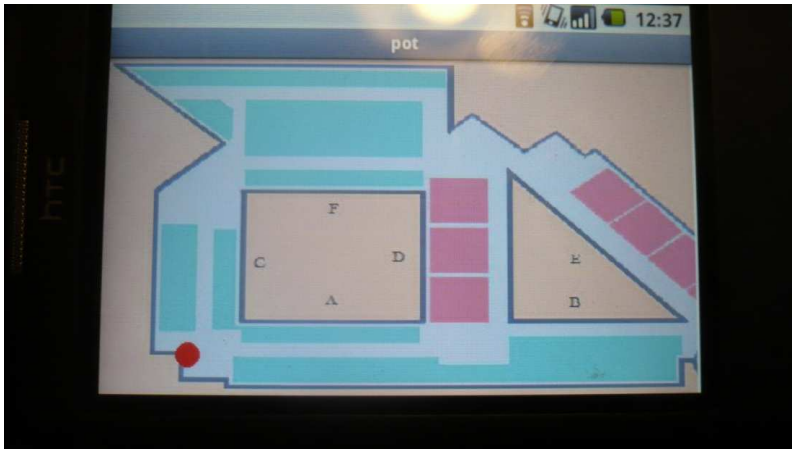


Figura 58. Vista del plano con un punto seleccionado.

En la figura 59 se ve el menú con las dos opciones y en la figura 60 se puede ver que la aplicación advierte que se han realizado todas las repeticiones de escáner programadas y que está listo para realizar la siguiente acción.

Figura 59. Vista del plano con un punto seleccionado y el menú seleccionado.

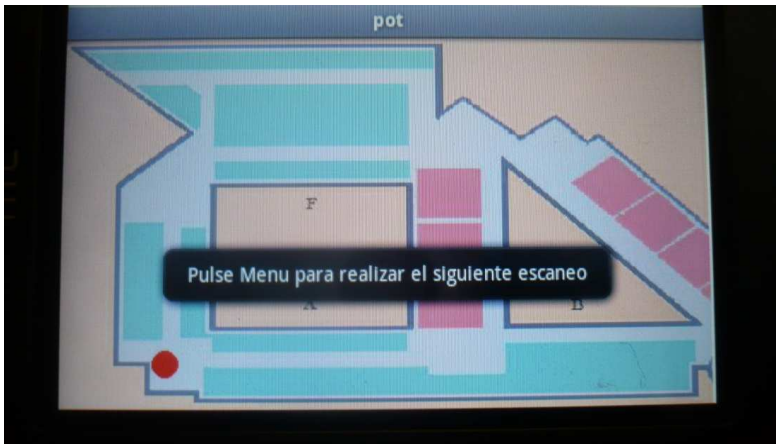


Figura 60. Aviso del fin del escáner.

- Localizador

Lo primero que se encontrará en esta aplicación es la pantalla inicial (Figura 61).

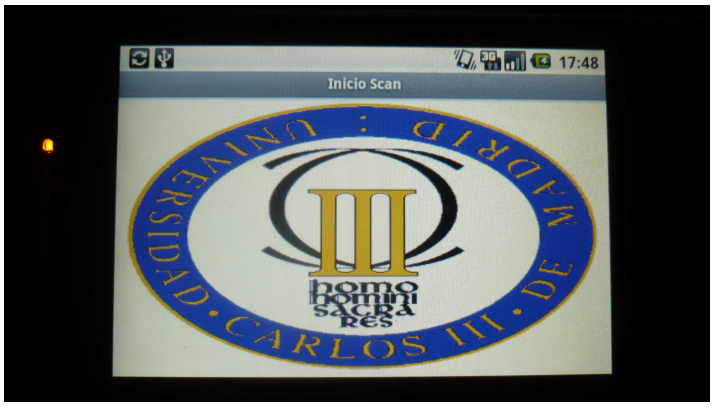


Figura 61. Pantalla Inicial Localizador

En la foto que se muestra en la figura 62 se ve reflejado una barra de progreso que informa de que se están produciendo las descargas de los archivos necesarios.

Figura 62. Vista de la aplicación cuando descarga los mapas.

Una vez finalizada la descarga de archivos aparece el plano del edificio y si se pulsa el botón de menú del dispositivo se puede comenzar a utilizar la aplicación, reiniciar su uso o finalizarlo (figura 63).

Figura 63. Vista del plano con un punto seleccionado y el menú activado.

Capítulo 5.
Conclusiones y líneas futuras.

5.1 Conclusiones.

En este proyecto se ha creado un prototipo de sistema de posicionamiento en interiores utilizando dispositivos con sistema operativo Android y la infraestructura de la red Wi-Fi del departamento de Telemática de la Universidad Carlos III de Madrid. Con este prototipo se consigue crear un mapa de potencias para que más tarde un usuario pueda conocer su situación en un entorno cerrado.

Como ya se ha visto el diseño de las aplicaciones se basa en la observación de las potencias de señal que llegan al terminal y la interpretación de éstas. Se ha buscado la sencillez tanto en la programación como en la interactuación del sistema con el usuario.

Se ha conseguido implementar el desarrollo completo del sistema con herramientas *Open Source* lo que lo hace accesible a todo el mundo. Además, se ha tratado de realizar un diseño eficiente en el que no interviniesen más dispositivos de los necesarios para abaratar costes, por ello se realizan todos los cálculos en el dispositivo móvil.

Los mayores problemas encontrados en la realización del proyecto vinieron derivados de las interpretaciones de las muestras en el entorno real, ya la red del edificio del departamento de Telemática no está dispuesta para este tipo de uso. Lo ideal para conseguir una localización perfecta sería poder realizar un estudio previo de la situación de los puntos de acceso para cubrir la mayor del edificio y que el dispositivo no sufriese confusiones a la hora de discernir la posición.

No obstante, gracias a las decisiones de diseño, como el algoritmo de cálculo de la posición basado en vector de potencias o la realización de los cálculos en el terminal o el uso de un fichero de texto sin formato para almacenar el mapa de potencias, realizadas durante la implementación de las dos aplicaciones desarrolladas en este proyecto, se puede es*table*cer con una buena precisión, en qué zona del departamento se encuentra un usuario. Además, en el diseño se ha buscado que fuese lo más general posible para que se pudiese aplicar a cualquier edificio sin necesidad de realizar ningún cambio en el *software*.

Se puede concluir que el sistema desarrollado en este proyecto difiere de los existentes en el mercado, ya que en el programa Ekahau todas las operaciones y cálculos se realizan en un servidor externo al dispositivo móvil, al contrario que en las aplicaciones aquí desarrolladas. Por otro lado, Place Lab no se basa en la posición del aparato móvil, sino en la posición de los puntos de acceso y además no tiene fase de calibración. En Herecast se realiza un aprendizaje de las zonas desconocidas sobre la marcha. Además, está preparado para entornos más amplios que los interiores de los edificios. Estas razones hay que añadir que el sistema operativo sobre el que se ejecutan las aplicaciones es Android.

Finalmente, a la hora de montar este sistema en otros edificios habría que realizar un estudio previo de cómo dividir los sectores para que el algoritmo tuviese el menor número de dudas posible a la hora de estimar la posición, consiguiendo así mayor precisión.

5.2 Líneas futuras.

Como línea futura estaría el hecho de mejorar la decisión de la localización. Para ello existen dos caminos a seguir.

El primero sería hacer un estudio de la mejor manera de distribuir los puntos de acceso en un edificio, para que se pueda cubrir la mayor cantidad de superficie con el menor número de puntos de acceso. Este camino sería el idóneo para los sistemas que se desarrollasen desde el principio incluyendo el montaje de la red Wi-Fi.

El segundo camino sería añadir una mejora al algoritmo de cálculo de la posición. Esta mejora consistiría en aplicar la teoría de grafos en el plano del edificio. Es decir, poner un valor a la distancia entre las zonas en que se divide la planta, de tal manera que la distancia entre zonas muy separadas en el plano fuese infinita y no pudiese decidir pasar de una zona a otra con esa distancia entre ambas. Si el algoritmo anterior fallase

se ha pensado utilizar *Tracking,* para estimar las posiciones en las que el algoritmo dude, quedándose con la última posición con la que no dudó.

Otra línea futura sería la de realizar estudios sobre las sensibilidades que tienen terminales distintos a la hora recibir la potencia de señal. En este estudio de las sensibilidades de los terminales habría que incluir factores de la red, como la sobrecarga, la red vacía, muchos usuarios pero sin hacer descargas, etcétera.

También sería necesario desarrollar una aplicación que permitiese a los usuarios la descarga de la aplicación y su instalación en el dispositivo móvil de la manera más sencilla y rápida posible. Además, se tendría que valorar la mejor opción a la hora de colocar esta aplicación: en un servidor y realizar una descarga como los mapas en esta aplicación, en ordenadores a la entrada del edificio, una descarga desde internet, vía bluetooth…

Una posible línea futura para la aplicación del Calibrador sería añadir un caso de uso que permitiese una rectificación en las medidas tomadas. Esta rectificación podría hacerse mediante un borrado de los últimos datos o una validación de los datos obtenidos antes de guardarlos en el fichero. Se propone el desarrollo de una biblioteca Java que maneje una base datos donde almacenar la información del mapa de potencias, permitiendo el borrado, adición y volcado en un fichero de los elementos que contiene.

Otra posible línea futura sería que cada vez que se conecte un usuario a la red de localización deberá rellenar un formulario con una serie de datos. Estos datos podrán usarse tanto para controlar el acceso de los usuarios, como para obtener información para realizar estadísticas sobre el rendimiento de la aplicación, sobre las zonas más visitadas, etcétera.

Capítulo 6. Presupuesto.

En el presente capítulo se presenta un cálculo aproximado del coste de este proyecto. Para la evaluación de dicho coste es necesario un desglose del proyecto en tareas, a través de la planificación del mismo y la evolución temporal que sigue. Cada tarea será cuantificada en función de su duración y los recursos consumidos, proporcionando de manera global un presupuesto aproximado del trabajo desarrollado.

6.1 Descomposición de tareas.

Una buena planificación de un trabajo es un aspecto clave para su posterior desarrollo con éxito. La identificación y descripción de cada una de las tareas a realizar, así como las dependencias que surgen entre ellas, puede ser complicado de estimar a priori, pero sirven como referencia acerca de la consecución o no de los objetivos que se deben cumplir.

Para la realización del trabajo expuesto se identifican cinco fases:

- FASE A: Documentación y estudio del estado del arte.
- FASE B: Implementación de la aplicación Calibrador.
- FASE C: Implementación de la aplicación Localizador.
- FASE D: Pruebas de las aplicaciones en el entorno real.
- FASE E: Documentación y memoria del Proyecto.

Cada una de las fases contiene tareas más breves, de objetivos más concretos. A continuación se pasa a hacer una breve descripción de las tareas realizadas en el desarrollo del Proyecto, indicando el esfuerzo temporal que conlleva cada una de ellas.

FASE A. Documentación y estudio del estado del arte.

TAREA A.1: Documentación del problema a tratar

- Descripción: Estudio de la aplicación Android y su entorno de desarrollo. Identificación de objetivos del Proyecto.

- Duración: 2 semanas.

TAREA A.2: Estudio teórico del estado del arte

- Descripción: Estudio teórico de los algoritmos y técnicas de posicionamiento en interiores basadas en radio frecuencia.

- Duración: 2 semanas.

TAREA A.3: Acercamiento a la herramienta AVD

- Descripción: Estudio de la herramienta AVD (Android *Virtual Device*). Se trata de la herramienta que simula un terminal.

- Duración: 2 semanas.

TAREA A.4: Ejecución de programas de pruebas.

- Descripción: Estudio de programas de ejemplo que ayudan a la compresión del funcionamiento interno de las aplicaciones, la interfaz gráfica y la comunicación entre bloques de un programa en Android.

- Duración: 2 semanas.

FASE B. Implementación de la aplicación Calibrador.

TAREA B.1: Implementación de la clase `WifiReceiver`.

- Descripción: Desarrollo de la aplicación encargada de realizar los escáneres, así como de la extracción de los datos de los mismos. En esta tarea también se desarrolla el menú de la aplicación para controlar cuando se desea realizar un escáner y para controlar el cierre de la aplicación

- Duración: 2 semanas.

TAREA B.2: Implementación de la clase `PintaPunto`.

- Descripción: Desarrollo de la clase encargada de obtener las coordenadas del plano expresadas en píxeles, así como de representar un punto en dichas coordenadas sobre el plano.

- Duración: 2 semanas.

TAREA B.3: Implementación de la clase auxiliar.

- Descripción: Desarrollo de la clase encargada de realizar todas las operaciones auxiliares de la aplicación: escritura en el fichero, la realización de los cálculos de la media de las señales en un punto y transformación de los resultados obtenidos en los escáneres a tipos de datos Java para su posterior manejo y ordenación.

- Duración: 2 semanas.

FASE C. Implementación de la aplicación Localizador.

TAREA C.1: Implementación de la clase `WifiReceiver`.

- Descripción: Desarrollo de la aplicación encargada de realizar los escáneres, así como de la extracción de los datos de los mismos. Se le añade la periodicidad a la hora de realizar el escáner. En esta tarea también se desarrolla el menú para controlar el cierre de la aplicación y el reseteo de la misma

- Duración: 2 semanas.

TAREA C.2: Implementación de la clase `PintaPunto`.

- Descripción: Desarrollo de la clase encargada de obtener las coordenadas del plano expresadas en píxeles, así como de representar un punto en dichas coordenadas sobre el plano. Se adapta la clase de la tarea B.2 para que obtenga las coordenadas del mapa de potencias.

- Duración: 2 semanas.

TAREA C.3: Implementación de la clase auxiliar.

- Descripción: Desarrollo de la clase encargada de realizar todas las operaciones auxiliares de la aplicación: lectura del mapa de potencias, la realización de los cálculos de la posición y desarrollo de los métodos encargados de realizar la descarga de los archivos en el dispositivo

- Duración: 2 semanas.

FASE D. Pruebas de las aplicaciones en el entorno real.

TAREA D.1: Realización de pruebas de la aplicación Calibrador.

- Descripción: Comprobación del funcionamiento de la aplicación en un entorno real. Tras las primeras pruebas se hace una recopilación de los errores y se procede a su corrección. Tras la corrección se realizan nuevas pruebas y se comprueba que el funcionamiento es correcto.

- Duración: 1.5 semanas.

TAREA D.2: Realización de pruebas de la aplicación Localizador.

- Descripción: Comprobación del funcionamiento de la aplicación en un entorno real. Tras las primeras pruebas se hace una recopilación de los errores y se procede a su corrección. Tras la corrección se realizan nuevas pruebas y se comprueba que el funcionamiento es correcto. Análisis de los parámetros adecuados (margen de sensibilidad, periodicidad…) para un buen funcionamiento.

- Duración: 1.5 semanas.

FASE E: Documentación y redacción de la memoria del Proyecto

TAREA E.1: Memoria completa del Proyecto

- Descripción: Redacción de la memoria del trabajo realizado, incluyendo una parte teórica sobre estado del arte y métodos empleados y una parte practica con descripciones detalladas de los sistemas desarrollados. Preparación de la defensa ante el Tribunal.

- Duración: 5 semanas.

6.2 Resumen de tareas

A continuación se incluye una tabla resumen de las tareas descritas con un cómputo de la duración total del Proyecto realizado:

Fase	Tarea	Duración (semanas)
A: Documentación y estudio del estado del arte	A.1	2
	A.2	2
	A.3	2
	A.4	2
B: Implementación de la aplicación Calibrador	B.1	2
	B.2	2
	B.3	2
C: Implementación de la aplicación Localizador	C.1	2
	C.2	2
	C.3	2
D: Prueba de las aplicaciones en un entorno real	D.1	1,5
	D.2	1,5
E: Documentación y redacción de la memoria del Proyecto	E.1	5
TOTAL		**28**

Tabla 15. Resumen de las fases y tareas del proyecto.

Por tanto, se han empleado 28 semanas para realización del sistema presentado.

6.3 Memoria económica

A continuación, se detallan los costes que conlleva la realización del Proyecto. Los costes se dividen en dos bloques: los costes materiales, de los componentes empleados en la realización, y los costes de personal, donde se detallan los honorarios de las personas que han participado, así como el tiempo que han invertido en el trabajo. Finalmente, se incluye la suma final de los costes.

6.3.1. Costes materiales

Los costes mostrados a continuación son los asociados al material necesario para la realización del Proyecto.

Concepto	Precio	Coste Amortizado
Ordenador Portátil	500	175
Dispositivo Móvil HTC Dream	490	200
TOTAL	**990**	**375**

Tabla 16. Costes materiales del proyecto.

6.3.2. Costes de personal

La Tabla 16 recoge los costes asociados a los honorarios de las personas que han participado en la realización del proyecto. El trabajo ha sido realizado por el ingeniero proyectista y dirigido por la tutora del presente PFC. La tutora hace las veces de Jefe de Proyecto, mientras que el ingeniero proyectista participa como Ingeniero durante el desarrollo de los sistemas y como secretario en la fase de redacción de la memoria del Proyecto. Las tareas de la A a la D son atribuidas al Ingeniero Superior, un total de 23 semanas, mientras que el secretario se ha encargado de la tarea E, de 5 semanas de duración. Por otro lado, el Jefe de Proyecto ha invertido un total de 40 horas.

Ocupación	Horas	Precio por hora	Importe
Jefe de proyecto	40	90	3600
Ingeniero	805	60	48300
Secretario	175	15	2625
TOTAL	**1020**		**54525**

Tabla 17. Costes de personal del proyecto

6.3.3 Costes totales

El coste total del proyecto se desglosa en la tabla 18

Concepto	Precio
Costes Materiales	375
Costes de Personal	54525
Subtotal	54900
I.V.A. (16%)	8784
TOTAL	**63684**

Tabla 18. Costes Totales

El coste total del proyecto asciende a *sesenta y tres mil seiscientos ochenta y cuatro euros*.

Leganés, 15 de Diciembre de 2009.

El ingeniero proyectista

Capítulo 7. Bibliografía.

[1] Haseman, C., (2008), *Android Essentials*, (1ª Edición), Apress, Estados Unidos.

[2] Murphy, M.L., (2009), *Beginning Android*, (1ª Edición), Apress, Estados Unidos.

[3] Hashimi, S.Y. y Komatineni, S., (2009), *Pro Android*, (1ª Edición), Apress, Estados Unidos.

[4] Ableson, F.W., (2009), *Unlocking Android*, (1ª Edición), Manning Publications, Estados Unidos.

[5] DiMarzio, J.F., (2008), *Android. A programmer's guide*, (1ª Edición), McGraw Hill, Estados Unidos.

[6] Meier, R., (2009), *Professional Android Application Development*, (1ª Edición), Wiley, Estados Unidos.

[7] Official WebSite Android, Consulta durante Segundo semestre del año 2009, <www.android.com>

[8]Anddev, Consulta durante Segundo semestre del año 2009, < www.anddev.org>

[9] Android- Spa, Consulta durante Segundo semestre del año 2009, <www.android-spa.com>

[10]Ekahau, Consulta durante los meses de junio y julio de 2009, <www.ekahau.com>

[11] Máster Oficial en Sistemas Electrónicos Avanzados, Fernando Seco Granja, *Departamento de Electrónica de la Universidad de Alcalá.*

[12] Place Lab, Consulta durante los meses de junio y julio de 2009, <www.placelab.org>

[13] Proyecto Fin de Carrera, *Sistemas de Localización de Dispositivos Móviles Basada en Wireless LAN,* Aurora Agudo de Carlos, Universidad Carlos III.

[14] Trabajo Fin de Carrera, *Sistemas de Posicionamiento Basados en Wi-Fi,* Aroa Carcavilla Sanz, Universitat Politècnica de Catalunya.

[15] ILS Sistemas de localización en interiores, Raúl Sánchez Vítores. Director de Proyectos de MIPSA.

[16] Herecast: Wi-Fi Location-based *Services*/802.11 Positioning *System*, Consulta durante los meses de junio y julio de 2009 <http://www.herecast.com>

[17] API Java 2 Platform SE v1.4.2, Consultado durante el segundo semestre de 2009, http://Java.sun.com/j2se/1.4.2/docs/api/

[18] Blueps: sistema de localización en interiores utilizando Bluetooth, Consulta durante junio y julio de 2009, <http://w3.iec.csic.es/ursi/articulos_gandia_2005/articulos/NC1/487.pdf>

[19] LOCADIO: Inferring Motion and Location from Wi-Fi Signal Strengths <http://research.microsoft.com/users/jckrumm/Publications%202004/locadio%20Mobiquitous%20distribute.pdf>

[20] CEDITEC , julio 2009, <www.ceditec.etsit.upm.es/localizacion.php?pagina=6>

[21] Artículo sobre la tecnología Wi-Fi, julio 2009, <www.ieee_infocom.org/2004/Papers/21_2.PDF>

[22] Pagina del Colegio Oficial de Ingenieros técnicos de Telecomunicaciones, Noviembre de 2009 <www.coitt.es/publicaciones/bit/bit 148/57-59.pdf>

[23] Actualidad-vnunet.es julio 2009, <www.vnunet.es/Actualidad/Noticias/Infraestructuras)Innovaci%C3%B3n/20050713004>

Apéndice A: Manual de instalación de las aplicaciones.

En este manual se va a mostrar de manera completa como instalar y configurar todas las herramientas necesarias para el desarrollo de aplicaciones en Android.

INSTALACIÓN DE ANDROID

1. Descargar Android del sitio oficial (http://code.google.com/android/download.html) y seleccionar el archivo que se indica en la imagen de abajo.

Version m3-rc37a

December 14, 2007 - Release Notes

Platform	Package	Size	MD5 Checksum
Windows	android_sdk_windows_m3-rc37a.zip	58 MB	5db5aea20a2c2f010baefc4b1091a575
Mac OS X (intel)	android_sdk_darwin_m3-rc37a.zip	54 MB	0b22e73fbd07b4af4009387afce3a37f
Linux (i386)	android_sdk_linux_m3-rc37a.zip	54 MB	41285beecc4f9926e6ecf5f12610b356

For more information on the SDK:

- Development Requirements
- Guide to Installing the SDK

2. El archivo se descarga dentro del mismo directorio donde se ha descargado el archivo de Java, se busca y se descomprime.

3. Descomprimido el archivo, este contendrá una carpeta llamada androd_sdk_windows_m3-rc37a (o similar dependiendo de la versión vigente). Esta carpeta se corta y se pega dentro de la unidad C del PC.

INSTALACIÓN DE ECLIPSE

1. En la página oficial de Eclipse (http://www.eclipse.org/downloads/) se descarga el archivo indicado en la imagen siguiente:

2. Una vez descargado, también se encuentra comprimido en .zip, se descomprime y se guarda la carpeta contenida dentro de la unidad C de la computadora.

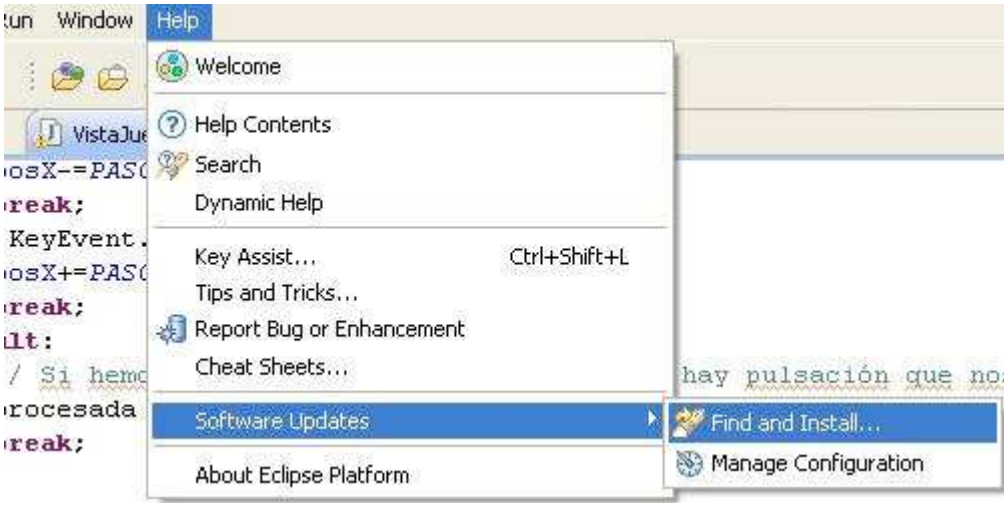
INSTALACIÓN DEL *PLUG-IN* DE ANDROID PARA ECLIPSE.

Si va a utilizar el IDE de Eclipse como su entorno para el desarrollo de aplicaciones de Android, puede instalar un *plug-in* llamado **herramientas de desarrollo de Android (ADT),** que añade soporte integrado para proyectos de Android y herramientas. El *plug-in* ADT incluye una variedad de poderosas herramientas que hacen la creación, funcionamiento y aplicaciones de depuración de Android más rápida y sencilla. Si no va a usar el IDE de Eclipse, no es necesario descargar o instalar el *plug-in* de ADT.

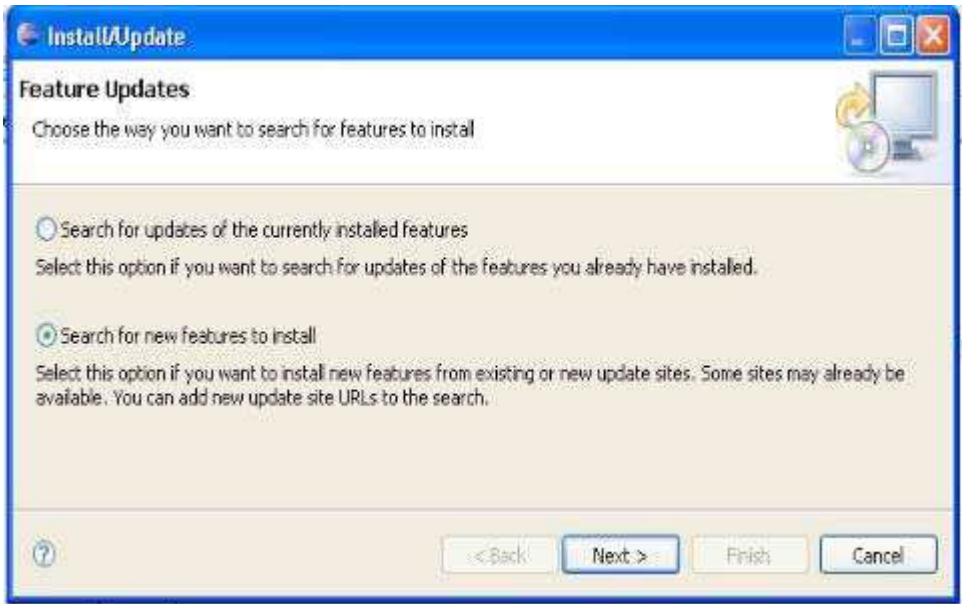
Ahora solo queda instalar el *plug-in* de Android para eclipse, para realizar esto hay que seguir los pasos siguientes:

1. Ejecutar el archivo eclipse.exe, se abrirá un diálogo donde preguntará la ruta donde guardará los proyectos que creados en eclipse. Por defecto apuntará al directorio de la sesión de usuario (usuarios Windows). Escoger la ruta deseada.

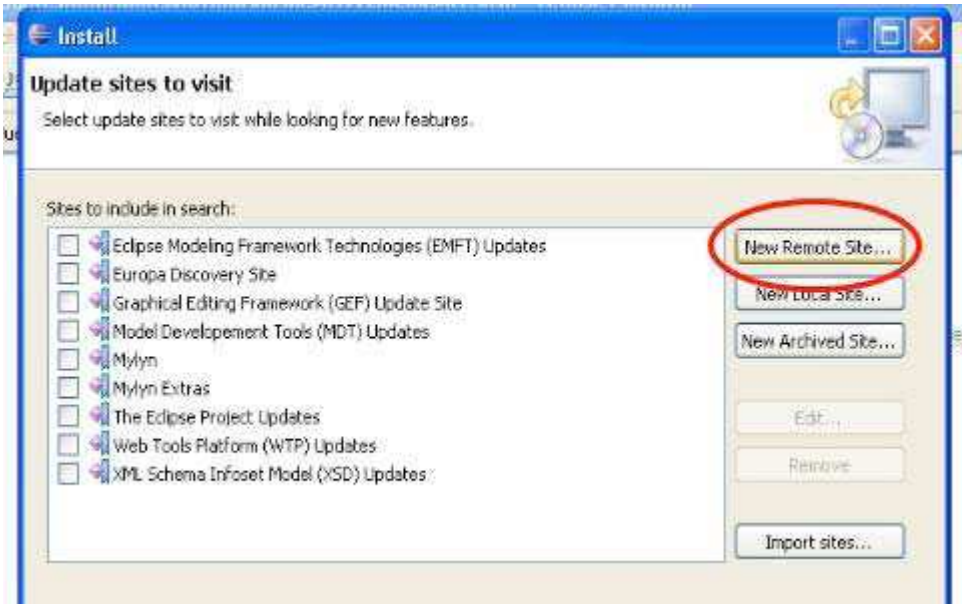
2. A continuación, instalar el *plug-in* de Android para eclipse, una vez ejecutado eclipse dirigirse al menú *Help* / *Software updates* / *Find* and *Install*.



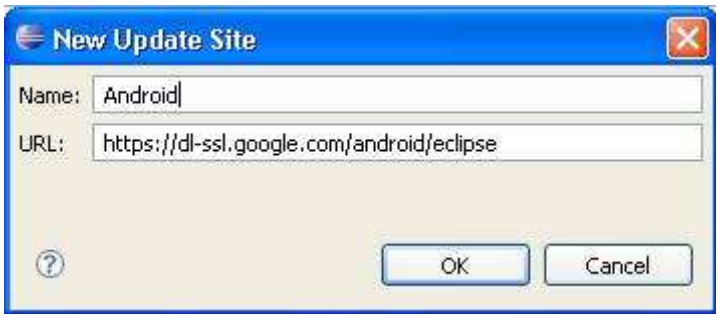
3. Se abrirá el diálogo siguiente donde se seleccionará la opción *Search for New Features to Install* y pulsar sobre *next*.



4. A continuación, se obtendrá el *plug-in* vía Internet desde el sitio oficial de Android, para ello en el diálogo siguiente se pulsará el botón *New Remote Site*.



5. Finalmente, en el siguiente dialogo se indica la url siguiente (https://dlssl.google.com/android/eclipse) donde eclipse obtendrá el *plug-in* y se le dará un nombre para identificarlo.



6. Ahora en la lista aparecerá entre las opciones Android, seleccionamos la casilla de Android y pulsar el botón *finish.* Se cerrará el diálogo y se buscará el *plug-in* en la ruta indicada, si todo sale bien se abrirá un nuevo diálogo donde se realizará la instalación del *plug-in* sólo debemos aceptar las condiciones y presionar ok. Y finalmente eclipse solicitará el reinicio del programa.

Apéndice B: Manual de Usuario

Aplicación Calibrador.

A continuación se van a explicar los pasos que hay que seguir para ejecutar correctamente la aplicación Calibrador.

1. Abrir la aplicación pulsando sobre ella en el menú.

2. Cuando aparezca el plano de la planta del edificio pulsar sobre la pantalla tantas veces como fuese necesario hasta ubicar en la posición correcta el punto de localización.

3. Pulsar el botón menú y a continuación aparecerán en la pantalla dos opciones:

   3.1 Repetir escáner si se quiere realizar el escáner de un nuevo punto.

   3.2 Salir de la aplicación si se desea finalizar el uso de la misma.

4. Cuando se pulsa Repetir escáner y éste finaliza se muestra un mensaje indicando esta situación. Una vez haya salido este mensaje se puede volver al punto 2 y repetir la secuencia hasta completar el mapa de potencias.

Aplicación Localizador.

A continuación se van a explicar los pasos que hay que seguir para ejecutar correctamente la aplicación Localizador.

1. Abrir la aplicación pulsando sobre ella en el menú.

2. Pulsar el botón menú y a continuación aparecerán en la pantalla dos opciones:

   2.1 Iniciar/Reiniciar escáner si se quiere comenzar o resetear la aplicación.

   2.2 Salir de la aplicación si se desea finalizar el uso de la misma.

3. Una vez se inicie la aplicación no se detendrá hasta que el usuario lo considere oportuno.

Apéndice C: Monitorización del nivel de batería.

A continuación se mostrará el programa utilizado para monitorizar el nivel de batería en las pruebas de la aplicación.

```
package pfc.bateria;

import android.app.Activity;
import android.os.Bundle;
import android.app.Activity;
import android.Content.BroadcastReceiver;
import android.Content.Context;
import android.Content.Intent;
import android.Content.IntentFilter;
import android.os.Bundle;
import android.widget.TextView;

public class bateria extends Activity {
        private TextView tvBatteryLevel;

   private BroadcastReceiver mBatteryInfoReceiver = new BroadcastReceiver() {
       @Override
       public void onReceive(Context context, Intent Intent) {
           String action = Intent.getAction();
           if (Intent.ACTION_BATTERY_CHANGED.equals(action)) {

               int level = Intent.getIntExtra("level", 0);
               int scale = Intent.getIntExtra("scale", 100);

               tvBatteryLevel.setText("Battery level: "
                       + String.valueOf(level * 100 / scale) + "%");
           }
       }
   };

   @Override
   public void onResume() {
       super.onResume();

       registerReceiver(mBatteryInfoReceiver, new IntentFilter(
               Intent.ACTION_BATTERY_CHANGED));
   }

   @Override
   public void onPause() {
```

```
        super.onPause();

        unregisterReceiver(mBatteryInfoReceiver);
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        tvBatteryLevel = (TextView) findViewById(R.id.tvBatteryLevel);
    }
}
```